

realtimepublishers.comtm

The Definitive Guidetm To

Exchange Disaster Recovery & Availability

Paul Robichaux



Jim McBee, technical editor

Chapter 5: Designing for Disaster Recovery and High Availability with Exchange	88
Exchange Architecture.....	88
Infrastructure Services	88
AD.....	88
DNS.....	90
Core Exchange Services	90
System Attendant.....	90
Information Store.....	91
Routing Engine	91
DSAccess	92
World Wide Web Publishing Service.....	92
SMTP	92
Ancillary Services.....	93
Points of Failure for Exchange	93
Physical Infrastructure Single Points of Failure	93
Communications and Internetworking Single Points of Failure.....	94
Hardware Single Points of Failure.....	96
Software Single Points of Failure	97
People and Process Single Points of Failure.....	98
Designing for Disaster Recovery	98
Backup and Recovery	99
What Do You Back Up?	100
How Do You Back Up?	102
Sizing and Timing: When and How Often Do You Back Up?.....	104
A Few Words About SAN-Based Backup.....	107
Storage Design.....	108
Service and Server Restoration.....	109
Designing for High Availability	109
Design Issues for Clustering.....	110
Design Issues for Replication and Failover	111
Designing for Planned Downtime.....	113
Summary	114

Copyright Statement

© 2005 Realtimedpublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimedpublishers.com, Inc. (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimedpublishers.com, Inc or its web site sponsors. In no event shall Realtimedpublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtimedpublishers.com and the Realtimedpublishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimedpublishers.com, please contact us via e-mail at info@realtimedpublishers.com.

[**Editor's Note:** This eBook was downloaded from Content Central. To download other eBooks on this topic, please visit <http://www.realtimepublishers.com/contentcentral/>.]

Chapter 5: Designing for Disaster Recovery and High Availability with Exchange

The preceding four chapters have talked about the fundamental technologies and processes that you have to apply when designing an Exchange system to improve its recoverability and availability. This chapter describes Exchange's infrastructure requirements as well as the business requirements for messaging availability and resiliency that have design consequences for Exchange. Rather than focus on the actual configuration steps, this chapter will describe your design options, how they work, and how to evaluate them to determine which settings are most suitable for your environment. Along the way, this chapter will draw on the introductory material from the previous chapters.

So far, you've learned a great deal about the underlying technologies that you can use to help improve the availability, resiliency, and service quality of your Exchange servers. It's time to move on to the next milestone—applying these tools and technologies to developing a serviceable Exchange design. We've already discussed some aspects of each technology that make it well- or ill-suited for Exchange; this is the appropriate point to widen the scope a little and look at the many other systems and services upon which Exchange depends so that you can broadly apply high availability, disaster recovery, and business continuity design principles to them.

Exchange Architecture

When you say Microsoft Exchange, what does it bring to mind for most people? A user will probably think first of Outlook, while an administrator would be more likely to think of the services Exchange provides—but those services are actually implemented as a collection of parts. Knowing what those parts are and how they interoperate is a key part of designing an effective protection strategy for Exchange.

Infrastructure Services

Technically, neither AD nor DNS are part of Exchange. In practice, though, both AD and DNS are critical parts of your Exchange implementation. Without them, your users' ability to use Exchange will be seriously limited.

AD

Exchange 5.5 maintained its own directory service that contained information about users, their mailboxes, and other mail-related objects. It was possible to have an Exchange user with no Windows account, and vice versa. Exchange 5.5 also allowed you to have a single Windows NT account that owned multiple Exchange mailboxes. With the advent of Exchange 2000, these features have changed, and Exchange has become tightly integrated with the Windows directory service.

Exchange 2000 and Exchange Server 2003 depend on AD for several critical functions. First and foremost is user authentication: when a user requests access to any mail object, the credentials the user provides are passed to an AD domain controller for validation. Such is the case regardless of which protocol the initial request used—RPC, HTTP, IMAP, POP, or NNTP. The domain controller uses Windows' standard authentication mechanisms to make a decision about whether the user should get access to the requested object; Exchange has nothing to do with the process other than initiating it and honoring its result.

The domain controller isn't the only important AD server role. Each domain controller in a domain contains a complete, writable copy of all the objects and attributes homed in that domain. However, AD was explicitly designed to support multiple domains in a single forest. To provide support for cross-domain lookups, Microsoft introduced the *global catalog* (GC) server. The GC contains a *partial* read-only copy of every object from every domain. Take user accounts as an example: the domain controller in your account's domain will have write access to every attribute associated with your account, while GCs elsewhere in the forest will have read access to a subset of your account attributes. This design makes it possible for a GC anywhere in the forest to efficiently answer questions about objects that live in other domains.

Exchange leverages this capability to use AD for user and resource location. Every mailbox-enabled user or InetOrgPerson object has an attribute called homeMDB defined. The value of that attribute is the mailbox database (and storage group and server) where that object's mailbox lives. Whenever you send a message to an Exchange recipient, the Exchange routing engine has to look up each recipient's homeMDB value to determine whether the recipient is local. Although Exchange caches these AD queries, if the queries can't be answered, mail delivery will fail. Individual Exchange components can also do AD lookups for various reasons; for example, the routing engine on a server will query to determine which computer is currently the routing group master. Most of these queries can be satisfied by connecting to any GC in the forest, although in some cases (notably distribution and security group expansion) a domain controller in the object's home domain may have to be involved.

Exchange also uses AD for storing forest-wide configuration data. You're already familiar with the concepts of AD domains, trees, and forests; you might not be as familiar with the *configuration naming context* (or just "config NC"). Objects and attributes stored in the configuration naming context are available to any server in the forest. Because Exchange uses the configuration naming context to keep track of which servers exist, what connectors link them to the outside world, and so on, servers in an Exchange organization can be in different domains, provided that they're in the same forest. (This design also leads to the limitation that you can have only one Exchange organization in an AD forest, because there's only one configuration naming context in each forest.)


DNS

DNS isn't glamorous, but it's a long-established core protocol for IP-based clients and servers, and AD is completely dependent on it for most functions. Every domain controller and GC server registers itself in DNS using a special type of DNS record called a service (SRV) record. Clients that need a particular service, such as Kerberos logon authentication, can query DNS for matching SRV records, then pick one based on IP subnet information. This method provides a quick, efficient way to find "nearby" network services. Exchange servers use this functionality at boot time to find domain controllers for logon and Group Policy application; they also use SRV records to locate domain controllers and GCs for answering requests for object attributes of various types (including distribution or security group membership, location of users' mailboxes, and the like).

Of course, Exchange depends on DNS in another way. Mail exchange (MX) records specify the IP address of a host that can accept email for an associated DNS domain. To receive Internet email, your DNS domain needs an MX record somewhere; to send outbound mail, your Exchange servers either need to have access to a DNS server that can look up MX records for destination domains or to be configured to use a smart host that can route mail on its own. One of the big challenges in setting up resilient Exchange services is making sure that the MX records for your domain are correctly adjusted in the case of a failover, and that you have an appropriate degree of redundancy in your DNS design.

Core Exchange Services

Exchange depends on the presence and function of a handful of core services. If these services don't work right, various aspects of your Exchange system will be degraded.

 For more detailed information about any of these services, see the "Core Exchange Server 2003 Services" section of the *Technical Reference Guide for Exchange 2003* at <http://www.microsoft.com/technet/prodtechnol/exchange/guides/E2k3TechRef/>.

System Attendant

The Exchange system attendant (SA) is the main control and management process for Exchange's other services. In this role, the SA serves a number of valuable functions, including providing an interface to Windows' clustering services, managing the Recipient Update Service, and scheduling and performing housekeeping tasks for other core processes. For example, the system attendant (via DSAcess, discussed later in this section) performs what Microsoft calls *topology discovery*; in other words, the SA is responsible for figuring out which GCs and domain controllers are available for Exchange's use, then choosing which ones will be used at a given point in time. The SA also runs the Mailbox Manager, handles publishing of free/busy information, runs the DS2MB threads that copy configuration changes from AD to the IIS metabase, and generates and updates the offline address book (OAB).

The SA is implemented as a Windows service (MSEExchangeSA); when it's stopped, none of the other core services will function (actually, some cannot even be started). The SA will normally run on every Exchange server in your organization.

Information Store

The information store (IS) service, `store.exe`, is what most administrators think of as the key core Exchange service. The IS provides client and system component access to mailbox and public folder databases; it implements all the transaction logging and data-integrity functionality and provides mechanisms for backing up and restoring data. The IS process itself implements one or more instances of the Extensible Storage Engine (ESE).

The IS must run on any server that offers access to mailbox or public folder data. Microsoft says that the IS is not strictly required for SMTP bridgeheads or front-end servers; because they don't have mailboxes, it is possible to turn off the IS on those machines. However, in the case of SMTP servers, the server won't be able to generate non-delivery reports unless there's a mailbox database mounted—in some cases, an SMTP bridgehead will stop routing mail if no mailbox databases are active on the server. If you plan to use a configuration like this, plan to test it carefully.

Routing Engine

Purists may not put the routing engine (RESvc) into the same category as the SA and IS components. However, if the routing engine isn't working on a given server, that server won't be able to send mail to other servers, inside or outside of the Exchange organization. Depending on your users' mail usage patterns, this limitation might not be a big deal; for most sites, though, losing the ability to send and receive mail to external recipients is a fairly big problem.

Like the IS and SA, the routing engine gets configuration data from AD. However, instances of the routing engine on different servers directly exchange routing data with one another. They do so in order to keep Exchange's link state routing table up to date on each server in an organization. There are two methods by which link state information is exchanged:

- During normal SMTP conversations between SMTP bridgeheads, the server can send the X-LINK2STATE SMTP verb to its peer; this verb is always accompanied by a binary blob of link state data that the receiving end uses to update its own local copy of the link state table. Servers within a routing group don't use X-LINK2STATE amongst themselves.
- Whenever a server notices a change in link state, it can notify its routing group master (RGM) by sending an update to the master's routing service, which listens for updates on TCP port 691. The RGM will then notify other servers in the routing group of the change by sending updates to them, also on TCP port 691.

When a routing service starts, it first retrieves a set of routing data from AD. All subsequent updates come from peer servers. That raises an interesting problem: if you want to flush a bad route out of your organization's link state table, you'll have to shut down the routing service on all servers in the organization and leave them off until *all* of them are shut down (or write a script that shuts them down for you). This task is easy enough if your servers sit next to each other, but in large distributed organizations, it can be quite challenging.

DSAccess

DSAccess is an interesting component; its job is to provide a standardized set of interfaces for *other* Exchange components to get data from AD domain controllers and GC servers. As you might expect, a busy Exchange server will make some of the same queries over and over. For example, every time a new message is submitted to the store, the message categorizer must decide whether the message should be delivered locally or remotely; to do so, the categorizer must query the directory to determine the location of the user's mailbox. Accordingly, DSAccess' first primary function is to serve as a cache for directory information; its second primary function is to perform initial topology discovery so that other system components will have access to GCs and domain controllers. After the initial discovery, DSAccess is also responsible for noticing changes in availability of GCs and domain controllers it has already discovered, then updating the local topology as necessary.

DSAccess is technically a DLL that runs in-process with the system attendant. However, this chapter lists it as a separate core service because it's possible to have trouble with DSAccess during times when the SA is otherwise functioning normally.

World Wide Web Publishing Service

It might seem odd at first to include IIS services in a discussion of core Exchange functionality. However, all of Exchange's Internet protocol support is built on top of the core protocol suite that ships with IIS. For example, the IMAP4 server ties into IIS' store driver, and it uses the *epoxy.dll* IIS driver for shared-memory communication with the IS. Besides IMAP, POP, and SMTP, the World Wide Web Publishing service is used for NNTP (which is fairly rare, actually); the main reason you should care about the health and status of the WWW service on your Exchange servers is that it's required for accessing Outlook Web Access (OWA), managing public folders via ESM, and using WebDAV clients like Entourage.

SMTP

Although the IMAP, POP, and NNTP services are implemented by Exchange using interfaces provided by IIS, the SMTP service is just the opposite: Exchange adds its own extensions (such as the link state routing capability mentioned earlier) to the basic SMTP service that ships as part of Windows 2000 and Windows Server 2003. The basic SMTP service is perfectly capable of accepting and transferring mail, but it lacks many of the features you would expect to see in a full-blown standalone mail server. Interestingly, several companies are making antivirus or anti-spam tools that can run on the basic Windows SMTP server, meaning that you can set up an SMTP gateway without using Exchange; when you do, though, you lose all of the Exchange management capabilities such as message tracking and queue management.

The impact of not having the SMTP service running is obvious—you can't exchange mail with users on other servers. In Exchange 5.5, the primary transport mechanism was an RPC-based X.400 implementation; in Exchange 2000 and later, SMTP is the primary means of moving messages between servers. If you want to stop message exchange between servers in your organization, stopping the SMTP service will do the trick.

Ancillary Services

There are a number of other services that Exchange uses or that implement additional services that are useful for Exchange management and control. For example, Exchange Server 2003 ships with connectors for message and calendar interchange with Lotus Notes and Novell GroupWise, the X.400 MTA allows interconnectivity with Exchange 5.5 servers or with foreign X.400 systems, and the Site Replication Service synchronizes the internal topology maps maintained by AD and the Exchange 5.5 directory service for mixed-mode organizations. These services aren't critical to normal operation, so they aren't specifically discussed in this chapter.

Points of Failure for Exchange

Before plunging into designing an infrastructure to provide high reliability and availability for Exchange, it's a good idea to take a step back and look at some of the specific single points of failure that can affect Exchange. Chapter 1 broadly discussed some causes of both planned and unplanned downtime, but it's important to understand specific Exchange failure modes in more detail.

Physical Infrastructure Single Points of Failure

Physical single points of failure such as electrical power represent a conundrum. On one hand, everyone understands the most common single points of failure and what their loss means—who hasn't had to put up with an unexpected power outage? On the other hand, these points of failure can be difficult to predict and expensive to guard against, so it's worth talking about them in some detail even though their obviousness may seem to make that unnecessary.

The first physical infrastructure single point of failure most people think of is electrical power. Without it, you won't be running Exchange, or any other computer-based applications. Loss of power is one potential problem, as is getting poorly conditioned power that exceeds normal voltage or phase or that contains spikes or drops in voltage. There is a whole class of related failure points, too: uninterruptible power supplies (UPSs) can run out of power or fail to cut over when needed, emergency generators can fail, and so on.

Many organizations have UPSs for their servers, which is a good practice. A few have emergency generator power, which can be a valuable way of ensuring electrical power. However, these systems add complexity, too. Consider sites that were using UPS made by APC who found their systems unstable after a digital certificate needed to run their administration software expired in July 2005—a clear case of the cure being as bad as, if not worse than, the disease!

Of course, there are other physical single points of failure that bear consideration:

- In companies that concentrate their computers into data centers, loss of cooling (either through power loss or cooling system failure) is always a concern. This situation is a great example of a single point of failure that you might not immediately be able to remedy; it's difficult to get backup air conditioning installed in commercial office space.
- Fire suppression systems are supposed to put out fires, or at least prevent their spread. They require maintenance, and if they go off (either accidentally or on purpose) it's likely that there will be interruptions in service associated with evacuating the spaces they protect and cleaning up afterwards—particularly if your servers are in a space protected by ordinary water sprinklers.
- Physical access is a potential single point of failure in itself. If you can't get access to the room in which your servers are located, you may still be able to remotely manage them, but you won't be able to handle mundane administrative tasks such as changing backup tapes or remotely rebooting a balky server in safe mode (although some vendors sell remote management tools that simplify this task).

The damage sustained by New Orleans and the Mississippi Gulf Coast during Hurricane Katrina provides a number of cautionary examples of how these single points of failure can have great impact. For most sites, it's not realistic to plan for extended operations under such conditions; just having a copy of your data is a good start, but it's not sufficient to keep your operations going when you have no place to operate and no computers to operate with—the two most likely outcomes of an extended or severe physical infrastructure problem.

Communications and Internetworking Single Points of Failure

Communications and internetworking are critical infrastructure functions that are all too often taken for granted. They also, unfortunately, are often out of your direct control as administrators. For example, suppose that you have a pair of T-1s connecting your primary data center to the Internet. Once those wires leave your building, there is almost nothing you can do to protect that connection from downtime, which is why many organizations choose to buy Internet service from multiple providers. Of course, you then have to worry about whether those Internet providers are using physically distinct connections to your building. Imagine buying service from two separate ISPs, then losing all communications one day when someone hits a utility pole near your building because both ISPs are sharing (or renting someone else's) optical fiber!

Communications and internetworking single points of failure to consider include the following:

- The physical layout of your connectivity. If you buy redundant T-1 service but the cables that connect you to the Internet provider's point of presence run through the same conduit, you're not getting much redundancy.
- The equipment that connects you to the rest of the world. If you have a single router, and it fails, you'll be disconnected; you can work around this possible point of failure by keeping a spare on hand.
- Protocols and services on which Exchange depends. The biggest example of this is probably DNS. If you're performing reverse DNS lookups on inbound messages, or using SPF or Sender ID to check the validity of senders' domains, messages can't be transferred when the DNS server is unavailable. Of course, if your Sender ID or SPF records are wrong, that will break your mail flow too.
- Any services, such as site-to-site VPNs, that you run to link remote sites to your central site. If the software or hardware that provides these links fails, the result at the remote site is indistinguishable from a major failure.
- The monitoring and control software you use to check the health and status of your connectivity.



Some of these potential failure points, such as where your cables go, are beyond your control. To mitigate them, you must be both realistic and creative. For example, the Church of Jesus Christ of Latter-Day Saints keeps a small stock of satellite phones on hand for its disaster recovery teams. Not only do these phones work when there is no cellular infrastructure available but they can provide IP connectivity (albeit at fairly low speeds).

Speaking of phones, don't forget your phone and voicemail system as a potential single point of failure. If you had an email outage combined with a failure of your in-building phone system, how would you communicate with the people you needed to reach to get things fixed?

Because communications and internetworking single points of failure can potentially interfere with all your business operations—not just messaging—your organization probably already has some kind of communications service plan that describes how to mitigate them. You should be familiar with that plan as well as aware of how any restrictions or changes it imposes might affect your ability to keep messages moving.

Hardware Single Points of Failure

Infrastructure single points of failure make many administrators nervous because these failure points are often out of your control. The topic of hardware failures, however, is familiar ground to most of us—after all, who hasn't had a hardware failure at some point in their computing career? Hardware sometimes fails, and hardware with many moving parts (such as disk drives) tends to fail more often. This reality will come as no surprise to administrators who have had disk drives, fans, and the like fail before. It's worthwhile, though, to classify hardware single points of failure into a few broad categories to give us a better idea of how to protect against them:

- The motherboard, CPUs, and support chips—These components don't usually have any built-in redundancy (especially on a single-CPU server); if one of these components fails, you should expect your server to abruptly quit working. There is nothing you can do to mitigate these single points of failure, other than keeping spare hardware handy to speed your recovery.
- Disk controllers and SAN host-bus adapters (HBAs)—Although you might install multiple HBAs in a server to use SAN multi-pathing, you can't generally install multiple controllers for other storage devices—including the system disk or direct-attached storage that your server is using. Caching controllers sometimes offer a means to remove the cache and reinstall it on another identical controller, but not all do.
- Network interface cards (NICs)—Depending on the NIC vendor you use, you may have either multiple NICs or a single NIC with multiple ports. Cheap NICs have a much higher failure rate than name-brand units, so you might want to keep that in mind as a possible mitigation step.
- RAM—RAM errors can be terribly annoying because they often manifest themselves as random misbehavior. Many servers offer the option to use ECC RAM, and you should definitely take advantage of this feature if your server has it.

You can't effectively mitigate motherboard or disk controller failures, at least not without using clustering. Some server vendors have started offering hot-swap PCI buses that allow you, in theory, to replace a failed NIC or HBA, although the Windows drivers for your particular brand of device may not be smart enough to support such swaps. Hot-swappable RAM is becoming more common, too, but it's still fairly expensive.

If you only have one of any component, it's a failure risk. For example, let's say that you have only a single tape drive that you use, over the network, to back up multiple servers. If that drive fails, how would you do a restore? If you had to restore multiple servers, you would need to do it sequentially, which might push your restore time beyond the in-place SLA or RTO. The same argument can be made for network appliances such as firewall or anti-spam devices, unified-messaging gateways, or anything else on which your network depends.

The best general way to mitigate hardware single points of failure is to buy good-quality hardware from a reputable vendor. The second approach is to add redundancy by duplicating capability—buying more tape drives, for example, or keeping a spare server with Windows installed ready to be converted to an Exchange front-end or an ISA Server firewall box.

You might be able to add redundancy to additional components—be sure to look for additional single points of failure outside the servers themselves. For example, consider a company that has dual NICs in all their servers, all of which were connected to the same Ethernet switch—that switch was a single point of failure in its own right, because its failure would have effectively disabled all the servers connected to it!

Software Single Points of Failure

As a software product, Exchange is remarkably reliable. It's a complex product, and so it's sometimes difficult to predict or understand its behavior; from a failure standpoint, though, it's quite robust. Whereas with hardware single points of failure, we're most concerned with the failure of individual *components*, with software it's more productive to focus on *services*, or the things that the software does.

Take antivirus scanning as an example. If your antivirus scanner fails, you can mitigate the failure in several ways. For example, you can reinstall the service to a different machine, either permanently or temporarily. One customer I worked with recently has an extensive implementation of Tumbleweed's content management products, with a total of about 10 servers dedicated to this third-party product. Tumbleweed's product has some built-in redundancy features, and the customer decided that during a failure, they could do without those services if necessary. Rather than spending money on clustering or other measures to improve the availability of an important but non-critical service, they focused their spending onto other areas.

Any software service whose failure impacts your business messaging operations falls into this category. For example, for organizations that are subject to the United States' Sarbanes-Oxley Act controls, the email compliance system may be critical because its failure can have unpleasant legal consequences. The same principle holds for companies that use customer relationship management (CRM) systems that integrate with Outlook and Exchange—if the CRM system fails, individual workers may not be able to get their jobs done even if the messaging system itself isn't directly at fault.

People and Process Single Points of Failure

The Gartner Group's research indicates that the biggest cause of downtime is administrators. Some estimates indicate that as much as 80 percent of system downtime is caused by people and process failures. This reality is at once scary and self-evident. Can you effectively mitigate people and process single points of failure? Absolutely! The first method of doing so is training. The more training and practical experience you can get for yourself and your fellow Exchange administrators, the better off your organization will be. Microsoft and a variety of other companies offer structured training (including instructor-led and online courses); some of them are expensive, while others are free. Microsoft offers an ongoing series of Exchange- and Windows-themed Web casts that cover troubleshooting, disaster recovery, and other Exchange administrative tasks.

There is also a lot of practical on-the-job training you can do, particularly if you use Virtual PC, Virtual Server, or VMware to set up learning Exchange environments that you can play around with without putting your production environment at risk. Virtualized environments are a great boon to professional development for Windows and Exchange administrators, as they let you test things that might otherwise be difficult, risky, or time-consuming to build out on physical hardware. (As a bonus, virtual machines are pretty portable, so you can easily take one home to play with on your own.)

The best way to get the right training, of course, is to practice your disaster recovery and business continuance processes. This practice has dual benefits. First, practice makes perfect (or at least better). Second, as you gain experience with disaster recovery and business continuity processes, you'll probably find areas in which you can improve your processes to speed them and make them more robust. Although this kind of training and planning isn't glamorous or exciting, it can make a huge difference when the chips are down, and you can do a surprising amount of learning and practice for free.

Designing for Disaster Recovery

Many fields have high requirements for availability—commercial passenger aircraft often implement doubly- or triply-redundant components for critical flight systems, nuclear power plants have designs that specify backups for backups, and the North American telephone network provides an excellent example of how highly available a large distributed system can be when you don't necessarily have to worry about how much things cost. How do you design your Exchange servers for disaster recovery? Most implementations must balance competing imperatives: save money, provide the ability to quickly recover from a disaster or outage, and do it all without unnecessary complexity and hassle on you and your staff.

Virtualization and Disaster Recovery, Business Continuity, and High Availability

What role should virtualization solutions such as Virtual Server and Virtual PC properly have in Exchange operations? The answer is a resounding “it depends.” Microsoft only changed its posture toward supporting virtualized Exchange servers with Exchange Server 2003 Service Pack 2 (SP2); for other versions, the company doesn’t officially support virtualized Exchange servers—thus, it’s reasonable to question whether something that Microsoft explicitly doesn’t support is something you should be building your disaster recovery processes around.

With the change in support policy, it might be more reasonable to consider using virtualized servers as part of your overall disaster recovery effort. Apart from their value in training and testing, virtual servers give you a quick way to create a needed server role or capability. Let’s say one of your front-end servers fails—you can quickly build a virtual machine and press it into service while you restore or replace the failed physical server. Whether this option makes sense will naturally depend on the size and complexity of your users’ workload, the number of users you have to support at once, and the details of your topology.

Apart from their value as “instant servers,” virtual machines give you a welcome degree of flexibility. A single physical host server, if adequately provisioned, can host several Exchange virtual machines, giving you a fairly straightforward way to do larger-scale restorations. Both Microsoft and EMC offer tools for copying physical servers’ contents to virtual machines, and there are lots of other unofficial (and unsupported) ways to get that data moved around. If you’re using Exchange 2000, one area you should probably investigate is using an on-demand virtual machine image as a way to quickly establish a recovery server; because Exchange Server 2003’s recovery storage group feature lets you mount a database on any other machine in the administrative group that has an RSG, this option is slightly less valuable.

One thing I *don’t* recommend considering—running your entire Exchange setup on virtualized servers. The benefits of being able to quickly move virtual machines between physical machines might seem worth investigating, but at the end of the day you end up with *more* complexity and overhead. If you have four physical Exchange servers, you need four Windows server licenses and four Exchange licenses, and you have four servers to maintain and patch. If you instead consolidate those servers onto virtual machines running on a single physical host, you then have *five* servers to patch, license, and maintain—generally not a move in the right direction.

Backup and Recovery

The preceding chapters of this book have explored generally backups and backup-related technical measures. This exploration should hopefully have given you an understanding of what technologies exist—now it’s time to talk about how you include those in your Exchange organization.

Let’s start with a couple of basic assumptions—most administrators have already internalized these, but it’s helpful to state them up front:

- You have restrictions on the amount of time your restores can take; this is going to be the ultimate driver for what your design looks like.
- You’re not backing up individual mailboxes; instead, you’re backing up storage groups and databases.
- You’ll take at least one full backup per week, along with either differential or incremental backups to provide coverage throughout the rest of the week.

These might seem like terribly basic assumptions, but they're important because they drive what your backup and recovery design looks like. For example, if your RTO is 6 hours, your backups shouldn't take longer than 3 hours. Your RTO may be longer or shorter; the important facts are that you *have* one and that meeting it will be the determining factor in what your final design looks like.

Planning a backup system requires several design decisions or elements that can be discussed as a unified set. You need to know:

- What you're going to back up—what volume of data, how it's divided into storage groups and databases, and what other data items you need to preserve.
- How you're going to back it up—the capacities and speeds of your backup hardware and software. Remember to check the restore speeds, too; some hardware can write data faster than it can read back.
- How much time you can allocate for backup and restore operations, and when you can schedule them.
- What explicit or implicit business requirements have an impact on your backup requirements.

What Do You Back Up?

The obvious answer to the question “What do you back up?” is “Your Exchange mailbox data.” In fact, that is the most important category of data that you need to preserve. The whole goal of most backup and recovery systems is to safeguard the mail data that your users need. There are two primary Exchange data sets that you have to back up:

- Transaction logs (*.log) give you a way to restore from the time of the backup to a point in time. You don't get to select exactly which point in time, unless you fiddle with the logs. Most often, log playback is used to restore as much data as possible; as long as you have all of the logs, in unbroken sequences, from the time of the backup forward, you can replay them to preserve all the transactions and thus recreate the database. As described in earlier chapters, different backup types may or may not truncate the transaction log files.
- The Exchange ESE database files (*.edb) and their counterpart streaming media (*.stm) files. Together, these files make up the contents of the Exchange mailbox databases on your servers. There are various tricks you can use to partially restore a database with one of these files when its partner is missing, but these tricks are a poor substitute for having complete, consistent copies of the files themselves.

The basic unit of Exchange backup is normally the storage group. Why? Because all the databases in a storage group share the same set of transaction log files. Although the ESE backup interfaces allow you to back up individual databases (and so do `ntbackup` and most third-party Exchange-aware backup tools), there's not much point in doing so—you would either have to not backup the log files or back up a separate copy for each individual mailbox database.

One additional nuance that most administrators don't realize is that the ESE interfaces allow you to back up more than one storage group at a time, given adequate backup hardware. Imagine a server with two storage groups, SG1 and SG2; each has two databases (SG1-M1 and SG1-M2, plus SG2-M1 and SG2-M2). If you're using a tool such as Symantec's BackupExec, you can backup SG1-M1 and SG2-M1 at the same time, provided that you have two tape drives or disks to write to in parallel. The ESE APIs let you back up all four storage groups in parallel, but only one database within each storage group can be backed up at a time—you can't back up SG1-M1 and SG1-M2 at the same time. Parallelizing your backups in this manner can cut your backup time dramatically, as long as you have enough hardware.

Of course, things are different if you have to restore an entire server and not just a single database or storage group. In that case, there are a couple of ways in which you can proceed. The old-school way is to reinstall Windows from scratch on the afflicted server, but as you can imagine, this tends to take a while. Other alternatives include using a system imaging tool such as Microsoft's RIS or Symantec's Ghost to quickly restore Windows to the server; most third-party backup tools provide an automated system recovery (ASR) function that boots a customized recovery CD, loads all the drivers necessary to restore from your backup media, and then does the restore.

Although your Exchange data is arguably the most important thing that you need to back up, don't forget the many additional data items that you must have for your Exchange organization to run. At a minimum, you'll need the ability to reconstitute all of the data in the configuration naming context of your AD forest, plus at least one domain controller for the domain that the server you're restoring is in. Because AD supports multiple-master replication, having any one of the domain controllers in the forest will suffice. Be extra careful if you're running Exchange on a domain controller (something that Microsoft recommends you not do, even though that's the whole point behind Small Business Server); in that case, you'll have to restore AD on the server before you can put it back into service as an Exchange server. If you're using anti-spam or antivirus tools that run on your Exchange server—make sure any data they generate (such as scanning databases or customized filtering rules) are included in your backup processes too, unless you don't mind recreating them by hand.

When Less Is More

Stand by for a controversial position: you can't back up everything, and you shouldn't try. This idea might seem to fly in the face of conventional thinking, but it's absolutely true. Unless you have an unusually large budget, you probably won't be able to deploy enough backup resources to back up every Exchange-related object and data item as often as you'd like. That forces you to make a choice between breadth of coverage and frequency of backup, and one way to make that choice is to elect not to back up everything.

Take Exchange public folders as an example. If you have more than one public folder server, you probably have more than one replica of a given public folder. Assuming your public folder replication is working properly (and yes, that's a big assumption in some cases), do you really need to back up every replica on every public folder server? The answer, of course, depends on what you use public folders for, how often the contents change, and a host of other factors that will vary from site to site. However, the basic principle remains: you might be able to get away without backing up every public folder database in your entire organization.

The same holds true for other replicated data items, such as NNTP public folders (which admittedly are not often found in Exchange sites) and even AD itself. This might smack of heresy, but the truth is that if you have more than one domain controller, you probably don't need to do daily full backups of every domain controller in the organization. Spacing the timing of these backups so that you back up an individual domain controller once or twice a week may suffice—if any individual domain controller fails, you can restore it from its most recent backup, and changes that have happened since that backup will be backfilled to the newly restored domain controller by the normal AD replication mechanism.

What does this approach give you? Well, assuming that one of the major constraints on your backup operations is the rate at which you can back up data to disk or tape, this frees up more of that backup capacity for use on the most critical data (most probably your Exchange databases). In addition, it simplifies your backup management processes by reducing the aggregate amount of data that you have to keep up with—and saves you long-term money by reducing the amount of media that you have to maintain in your archival storage system.

How Do You Back Up?

Most administrators naturally think of tape as their primary backup medium; tape backup has been the industry standard for almost 50 years now, and tape capacity and speed have increased steadily over that time. For several hundred dollars, you can buy a DLT-IV drive that uses 1/2" tape to back up between 35 and 70Gb to a single tape (the exact amount depends on how compressible the data are) at a rate of several Gb per hour. Of course, you can certainly spend more money; companies such as ADIC and StorageTek (now part of Sun Microsystems) will be happy to accept your multi-hundred-thousand-dollar check for one of their large autoloading tape libraries. Capacity and speed are the two measurements by which tape systems are normally measured, but I like to include cost per Gb as a measurement too. Why? Because when you look carefully at per-Gb costs you may find that tape isn't the best solution for your short-term backup requirements.


Tape has its advantages: it's a well-understood and mature technology that has been widely used and deployed on a wide variety of platforms for a long time. Media management, rotation, labeling, and library management are well-integrated with most third-party backup software packages, and tape offers a fairly stable and cost-effective solution for long-term archival (provided you don't lose or damage the tapes while they're in storage!)

However, it also has its disadvantages. Speed is probably first; the linear nature of tape means that the only ways to speed up the process is to run the tape faster (which introduces a host of interesting physics and engineering problems) or to put more data on it (making the tape more vulnerable to head misalignment or physical or magnetic damage). Tape I/O speeds have certainly increased over the past 40 years, but at a much slower rate than the I/O speeds offered by disks. This obviously has an impact on your backup and recovery design; one key factor in designing your backup scheme is how much data you can actually back up in the amount of time you have allotted. Restore speed is arguably more important than backup speed, of course, and most designs attempt to optimize restore speed at the expense of cost. There's actually a third way to speed up tape: parallelize your backups, which I mentioned earlier. Some third-party backup utilities allow you to stripe data across multiple tape drives, basically making a RAID-1 array of your tape drives. As you might expect, this method gives you terrific throughput, but if you lose one tape in the stripe set—or one tape *drive*—you can't restore. That makes striping inappropriate for most backup systems.

A bigger problem is reliability. Although I've never seen convincing large-scale statistics, several analyst firms have reported that customers they surveyed reported that more than 40 percent of restore attempts failed—not a very comforting statistic given that backup systems are supposed to be the last line of defense against abject failure. (In a recent high-availability-themed Web cast I did, two thirds of the audience reported a serious or moderately serious restore failure—even worse numbers!) These failures can be caused by failed media, missing or damaged media, or problems with restore procedures. These problems can take on a number of forms that aren't as important as the fact that your design should take into account the likelihood of having problems. Don't depend on a single server to do restores for multiple Exchange boxes, and don't rely on a single tape drive—have backup drives for each of your production drives so that you can restore as fast as possible during an emergency. You should also be prepared to deal with the unfortunately common circumstance of finding out—the hard way—that your backups haven't been working right for a while. (Of course, good monitoring and test procedures will ensure that you find out about backup failures early enough to recover from them!)

For these reasons, many administrators are turning toward using disk-to-disk backup. D2D systems take the stream of ESE data produced by the ESE backup interfaces and serialize it into a single file that's written to disk. Once it's on disk, the backup file can be copied, moved, or written to tape just like any other file. The obvious advantage of this approach is speed: D2D transfers (especially between devices on different controllers) can be an order of magnitude faster than relatively poky tape transfers. This speed boost can dramatically cut backup and restore times, making it possible for you to back up more often and restore faster.

D2D systems have their drawbacks, too, chief among which is the difficulty of storing multiple copies of your Exchange data on the disk. If you have 150GB of Exchange data, keeping a week of daily full backups around will cost you 1050GB, or just over a terabyte, of disk space. Furthermore, disk isn't a good long-term storage mechanism for most situations because it's too expensive to leave languishing in a storage vault somewhere. For these reasons, D2D is often combined with a tape stage: initial daily backups are done to disk, kept around for a day or two, then written to tape for longer-term storage. This so-called disk-to-disk-to-tape (D2D2T) strategy neatly handles the chief problems with both tape- and disk-based backup systems.

 For a look at how Microsoft uses D2D2T in their own operations, see the “Backup Process Used Single Point of Failure with Clustered Exchange Server 2003 Servers at Microsoft” white paper at <http://www.microsoft.com/technet/itsolutions/msit/operations/exchbkup.mspx>.

Sizing and Timing: When and How Often Do You Back Up?

Conventional backups aren't instantaneous. Thus, you must think about three things: when you're going to start them, how long they'll take, and how much data will you thus be able to back up within your time window. Let's take these concerns in reverse order.

In the old days (say, 5 years ago), most Exchange servers had a small enough amount of data so that you could do one or more full backups onto a single tape, although some servers might have required the use of DLT or LTO tape rather than the less-expensive, smaller-capacity Travan standard. As time has passed, though, such is no longer always true. Even for shops that have less than a single tape's worth of data, the time it takes to fill that tape is still a concern. The most common way to address this concern is to size your servers using a fairly straightforward process:

1. Determine how many mailboxes you have, both per server and in total.
2. Calculate the maximum expected average mailbox size. If you're using mailbox size limits, this is simply the size limit. If not, you can dump your existing mailbox sizes to an Excel spreadsheet (see <http://www.exchangecookbook.com> for scripts to help with this) and derive an average.
3. Identify the sizes of every data set that needs to be backed up. For our purposes, it's probably OK to stick with storage groups instead of databases as the unit of backup.
4. Identify your RTO, then cut it in half. Why? Remember that restores always take longer than backups; Microsoft's planning figure is that restores take twice as long. This may seem unnecessarily conservative, but I've found it to be a pretty good estimate for most sites. Let's call the resulting time your backup window.
5. Identify the amount of data you can back up within the backup window. Obviously to do so, you should know how fast your backup system performs in actual use—a good excuse to do some testing!

You then know how much data you have to back up and how much data you can back up in the allotted time. For servers whose data can be backed up within the backup window, you're in good shape. However, unless you have a single server, you might have more data on some servers than others, so you're not quite done with the sizing process:

- Calculate the maximum size you can accept for each storage group. This is simple math: divide the number of gigabytes you can back up during your backup window by the expected mailbox size to determine how many mailboxes you can have per storage group.
- If you have storage groups or servers that are over the size limit, move mailboxes around until every storage group is below the threshold size.

An example might help clarify things. Let's say you have an Exchange organization with four servers, totaling 3000 users (as a simplifying assumption, let's say that all the servers are equally sized). Each server has one storage group with two databases. You use a 150MB mailbox size limit. You have a 12-hour RTO. Thus, your backup window is 6 hours. Your shared backup subsystem can handle 12GB/hour. In order to restore a single server's worth of data, you would need to move $750 \text{ users} * 150 \text{ MB} = 113\text{GB}$ of data. However, during the allotted backup window, you can move at best about 72GB of data. Your choices are then to:

- Spread the data across more storage groups on the same server, then back them up in sequence. This method allows you to have multiple backup windows; splitting the 750 users on one server into two storage groups gives you about 65GB of data in each storage group. Back up the first storage group from 7PM to 1AM, and the other from 2AM to 8AM. (You still must beware the online maintenance window.)
- Spread the data across more servers. This option seems counterintuitive, as most sites are always looking to *reduce* their server counts because doing so saves money. This is just another tradeoff, though: server cost versus recovery speed. Adding one more server means that each server would then have 500 users with a maximum of 75GB of data; because not every user would hit the maximum mailbox size, this would probably work.
- Get a faster backup system. Adding a second tape drive to each server would allow you to back up two storage groups in parallel; you could also upgrade to a faster tape system, or just use disk-to-disk backups to capture a nightly full backup and then drop it to tape at a convenient time during the day.

I mentioned the online maintenance window, and it's important to remember it during your backup planning. Exchange performs a number of important housekeeping tasks, including online defragmentation, aging out items in public folders, and removing previously deleted mailbox items whose deleted item retention period has passed. By default, this process starts at 1AM local time (although you can change this with the Customize button on the Database tab of an individual database's properties dialog; see Figure 5.1). More importantly, if this process is running when a backup of *any* database starts, online maintenance stops (or, more precisely, pauses).

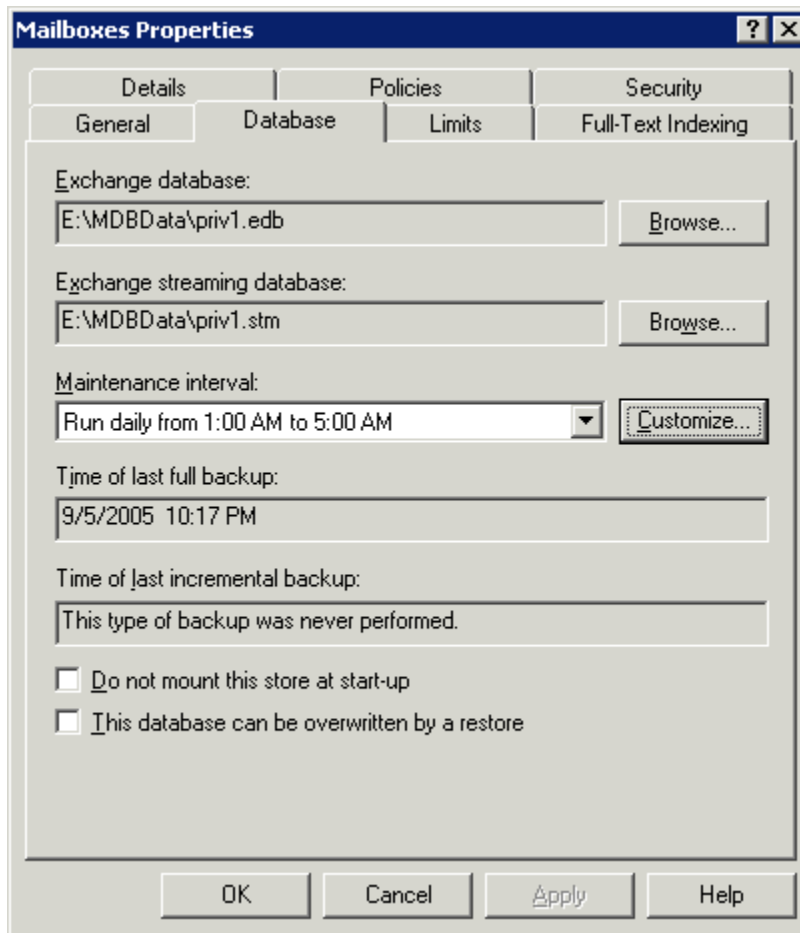


Figure 5.1: Set the database maintenance interval using the Customize button.

This is a problem because the maintenance tasks have to run to completion to keep the database in good order. Let's say you have the default maintenance window set for a database, and you're doing nightly backups beginning at 2AM. The maintenance task will start at 1AM, but it won't be able to finish because when the backup starts, it has to pause. If the backup finishes before 5AM, the online maintenance task will resume, but it will have to stop at 5AM. You see the problem here: if the task doesn't run to completion, it will have even more work to do the next time it runs—which means it will take longer. As time passes, the task will fall further and further behind.

To fix this, you have to ensure that your online maintenance and backup windows don't conflict. Your backup either has to complete before online maintenance begins, or online maintenance should start after the backup's done. If you can, let online maintenance run first; that way, when you complete the maintenance period, you'll be backing up data after the store has had a chance to complete its work.

A Few Words About SAN-Based Backup

Many companies are examining the possibility of using SAN-based backups, either with Microsoft's Volume Shadow Copy Service (VSS) or vendor-specific solutions. For planning purposes, you can essentially treat these like instantaneous D2D backups: "instantaneous" because they can be much faster than conventional D2D backups and "disk-to-disk" because they generally work by copying their data to an equivalently sized disk volume—if you have 200GB to back up, you'll need at least 200GB of free space on your SAN. SAN-based backup systems offer a way to quickly capture point-in-time copies of your Exchange data, but you're still faced with the requirement to have "spare tire"-style backup to protect you in case the SAN-based copy isn't available when you need it.

It's important to understand the two separate types of SAN-based backups. Microsoft defines them in the following manner in the "How Volume Shadow Copy Service Works" white paper on TechNet

(<http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/TechRef/2b0d2457-b7d8-42c3-b6c9-59c145b7765f.mspx>); I've copied their descriptions verbatim to make sure that the description here doesn't miss any nuances:

A clone is a full copy of the original data on a volume. You can create a clone through either software or hardware mirroring. Clones remain synchronized until the mirror connection is broken for the shadow copy. From this point forward, the source data and the shadow copy volume are independent. The original volume continues to take application changes, while the shadow copy volume remains an exact read-only copy of the original data at the time of the break. Hardware vendors offer different hardware-based implementations (sometimes called split mirrors, snapshot mirrors, or clones) for creating identical images of volumes that can be used for online backup, application development, and testing.

The copy-on-write method creates shadow copies that are differential rather than full copies of the original data. Like the clone method of creating shadow copies, the copy-on-write method can produce shadow copies using either software or hardware solutions. This method makes a copy of the original data before it is overwritten with new changes. When a change to the original volume occurs, but before it is written to disk, the block about to be modified is read and then written to a "differences area", which preserves a copy of the data block before it is overwritten with the change. Using the blocks in the differences area and unchanged blocks in the original volume, a shadow copy can be logically constructed that represents the shadow copy at the point in time in which it was created.

Clone backups are preferable for most uses, as they're exact copies of the data and are kept current right up until the clone is separated from the original. As long as the original database is consistent, or can easily be made consistent, clones fill the bill. Copy-on-write copies are not as suited to use with Exchange, although they work well in environments in which what you really want is to take the changes recorded in the differences area and copy them to a remote SAN unit.

Storage Design

As I mentioned in earlier chapters, your storage design has a strong effect on both the performance and recoverability of your Exchange system. By this point, you've probably had a chance to consider the different RAID levels and choose one that's appropriate to your budget and requirements—hopefully, you strongly consider using RAID, as it offers valuable high-availability protection.

The more interesting storage issues from a disaster recovery perspective are simple and straightforward, and most revolve around either speed or redundancy:


- Whenever possible, put your tape drives or backup disks on their own I/O channels. This setup helps maximize throughput as you read or write backup data, which both contributes to your overall backup speed and helps protect you against otherwise-problematic failures of individual channels (or, more likely, cables or other paraphernalia).
- Make sure you have a sufficient amount of disk space to run Exchange's recovery utilities, should you need them. For example, `eseutil` requires as much as 110 percent free space to perform an offline defragmentation of a database (that is, if your database is 80GB, you may need up to 88GB free to defragment it).
- Remember that arrays aren't generally interchangeable between controllers (unless they're identical down to the firmware version), and that drives from one array can only be reused in another if they're erased. No SAN of which I'm aware allows you to pull a drive from one enclosure and use it, data intact, in another similar SAN.

Service and Server Restoration

So far, most of what we've been concerned with involves backing up and restoring your Exchange data. However, simply restoring your Exchange data is semi-meaningless if you can't efficiently restore the services themselves, by which I mean Windows, AD, and Exchange. When it comes to service restoration, there are four basic scenarios:

- You have to restore the Exchange data, but your Exchange server is still working. This will be the case if you have to reload a damaged or corrupt database, or if you have to restore a database to an Exchange Server 2003 recovery storage group.
- You have to restore Exchange *and* Exchange data, but the underlying Windows installation is OK. This will happen when you have to install Exchange on a server to replace a failed server.
- You have to restore Windows, then Exchange and Exchange data. This happens when you break out a replacement server from its shipping box, for example, or in other cases (say, repurposing a Linux server) where there's no usable Windows installation.
- You have to restore Windows on an AD domain controller. This requires a fair amount of specialized knowledge, as you must first restore Windows and then use the directory services restore mode (DSRM) to restore the directory data (and then wait for AD changes to replicate back from other domain controllers).

There are a variety of ways to reinstall or repair Windows on a failed server; for Exchange data, you need to be able to execute conventional backups and recoveries as described in this chapter and the next one.

 Chapter 6 explores reinstalling Exchange on a server (with or without reloading databases).

Designing for High Availability

The earlier parts of this chapter have focused on designing for disaster recovery. Now it's time to take a different path and examine some of the design considerations in systems that are being designed primarily to deliver better availability.

First, let me begin by drawing the distinction again between service and server availability. There are several parts of Exchange in which you care about the *service* and not the individual server; examples include front-end servers and SMTP bridgeheads. For these roles, you can provide availability by using two common techniques: DNS round-robins and network load balancing (NLB).

The idea behind round-robin DNS is simple. The DNS standard includes provisions for having multiple A or MX records associated with a given name. The DNS server is responsible for giving out an equal number of replies using each record; if you have five A records registered for your SMTP server, then 100 consecutive queries for that server's DNS name should return each of the A records a total of 20 times. This is a simple way to provide redundancy, and thus enhanced availability, for simple services that don't maintain state—like inbound or outbound SMTP, IMAP, or POP. However, the round-robin mechanism has no way to know that a server isn't available—if one of your five servers fails, the DNS server will still happily hand out its address 20 percent of the time.

Network load balancing is slightly more sophisticated. The idea here is that network traffic is directed to a virtual resource (shades of clustering!), and that some other mechanism is used to actually distribute the traffic to one node at a time. This can be accomplished by the Windows NLB service, or by a hardware device such as those made by Cisco and others. NLB systems work well for services such as OWA that have to keep state information around so that an ongoing client session can be maintained.

However, both round-robin and NLB protect you only against failure of individual service providers. These tools match requests to services; they don't do the same kind of detailed status checks that Microsoft's clustering implementation does. As a result, even though they add redundancy, they're not sufficient by themselves. Let's say that you have two front-end servers in an NLB group. If the Exchange services on one of those servers fail, NLB will happily keep sending work to that server—which can't handle it because its Exchange services are offline.

Design Issues for Clustering

I've talked at length earlier about clustering and its pros and cons. From a high-availability standpoint, clustering is a valuable way to increase the availability of your servers by making them more resilient against certain types of problems. However, you have to counterbalance the increase in availability that you get against the added expense and complexity that you have to suffer. Clustering may make sense for your organization when you consider it as an overall part of your high-availability design. However, I don't recommend spending any money on cluster implementation until after you've built out a solid disaster recovery design *and* made sure that you have the necessary design and operating expertise to handle clusters.

From a design standpoint, if you're planning to use clustering, you need to incorporate adequate network connectivity between the cluster nodes. Remember that the cluster interconnect used for heartbeat traffic between the nodes should be a private network segment; for a two-node cluster, using a crossover cable eliminates the potential that an intervening hub or switch might fail. Most cluster designs use two separate NICs: one for the interconnect and one to connect the cluster to the LAN. This setup is a wise design; although you can accomplish the same outcome by using a single multi-port NIC or even NIC teaming, you may accidentally introduce a single point of failure without meaning to do so.

If you're using SANs with your cluster, you also need to consider how your cluster nodes will communicate with the SAN. The cluster nodes will each have at least one HBA connected to the SAN; for redundancy, you might want to use two HBAs per node, ideally connected to the SAN through different physical paths. Whether this redundancy provides benefits for you depends on the SAN and HBA vendors and how well their drivers implement SAN multi-pathing. It's critical to test multi-pathing to ensure that it behaves correctly when a SAN interconnect changes state. If you're thinking of using SAN management software (either from your SAN vendor or a third party), be sure to carefully consider its stability and quality because those factors will have a huge impact on the level of availability your SAN adds—or subtracts.

Design Issues for Replication and Failover

The most obvious design issue for replication is bandwidth: how much of it you have, what latency it imposes, and whether those criteria are acceptable. What “acceptable” means in this context varies; Microsoft recommends that you minimize write latency, but if you're using asynchronous replication, you might wonder whether latency still matters. The answer is that latency does matter, although increased latency won't have a great impact on the performance of your production server. Latency *will* affect how long it takes to get updates from the source server to the target server, and that, in turn, affects how “fresh” the data at the remote site will be.

Adding bandwidth may or may not be an option for your organization. However, software replication products such as XOsoft's WANSync or NSI's DoubleTake generally allow you to control replication in various ways, including scheduling replication by time of day (see Figure 5.2) or date, or triggering replication updates either manually or when certain events take place. These features allow you to get by with less bandwidth because you're not replicating everything all the time.

Another useful way to reduce the amount of bandwidth consumed is to remember that replication works in conjunction with conventional backup and recovery solutions—not as a complete replacement. If you set replication to happen at preset intervals and perform local backups at intervals *between* replications (being sure to exclude any on-disk files you create as part of the backup), you can move the recovery point objective (RPO) to whatever level is appropriate for your needs.

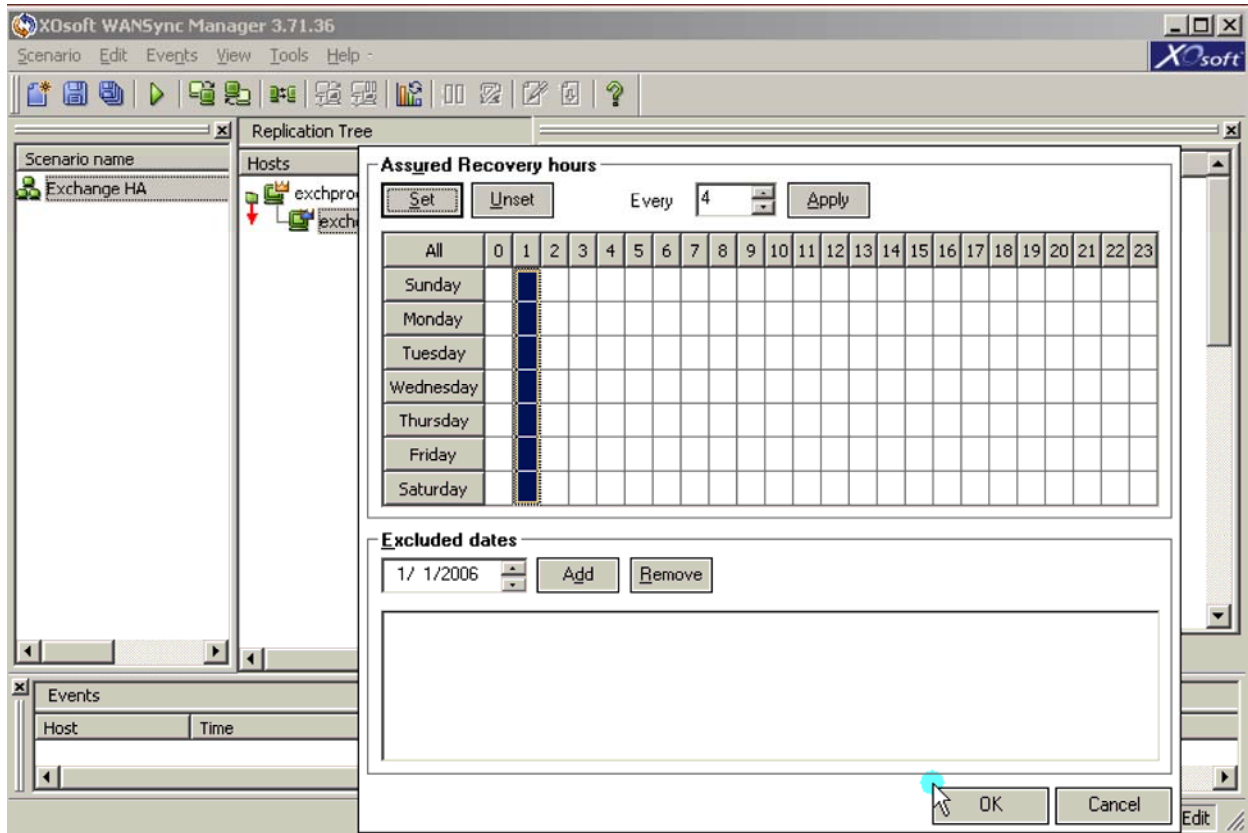


Figure 5.2: Setting the assured recovery hours in WANSync lets you specify when replication occurs.

Failover requires a separate set of design decisions, but they're mostly focused on when and why you fail over, not so much on how you do so. The reason is that Exchange requires a fairly standardized set of operations if you want to fail over client connections to another computer. Whether you're using Windows clustering (which does all of the heavy lifting automatically) or a third-party solution that automates some or all of the failover actions, you'll be doing the same things. It's important for you to specify under what conditions you want failovers to occur (for example, you probably don't want a failover to happen every time you have a 5- or 10-minute outage). It's also important that you test the failover procedures you're using, particularly if you're using older versions of Outlook. If you don't have a quick way to tell users that they should quit and relaunch Outlook, for example, your Help desk is likely to be inundated with calls every time you have a failover.

At least once a year, you should plan to fail over operations to your remote site *and leave them there* for a few days or so. This practice is a great way to test your failover procedures and replication in a real-world environment—much like switching your electrical service over to your building generators for a few hours each month. To minimize risk, you can, of course, do this over holidays or other periods when you have reduced traffic, but it's important to design regular exercises into your deployment plan.

Designing for Planned Downtime

You can invest a lot of time and money in preparing for emergencies that never come. That doesn't mean that you shouldn't plan and prepare for them, but as you do so, don't forget the one circumstance that's *guaranteed* to come: service packs and hotfixes for your Exchange servers.

Over the past 2 or 3 years, Microsoft has put a great deal of effort into standardizing the way that its service packs and hotfixes are packaged and documented. That means, for example, that when you get a hotfix, you should immediately be able to tell which services (if any) have to be stopped to install it and whether a reboot is required afterwards. Having this information is handy, because it helps you plan, but the information alone isn't enough—you'll still have to have a plan for getting those hotfixes or service packs installed on your servers.

Alternative #1 is what IT administrators have been doing for a long time: installing patches and fixes at times when users aren't using the system. This might sound reasonable until you consider the nature of email systems: there are very few times when users aren't sending or receiving mail, and most of those (Christmas Day, during the Super Bowl, Labor Day weekend) are times when you probably don't want to be in the office.

Alternative #2 is to set aside times for planned maintenance when it won't greatly inconvenience users. Many organizations set a planned maintenance period at a fixed time each month: say, the third Saturday of the month between 10PM and 4AM. This option is less inconvenient for both administrators and users, but in opposite directions: admins want to do their planned maintenance during normal work hours, and users want it done at some time when it doesn't interfere with their work.

You can probably guess where this is leading; alternative #3 is to figure out a way specifically to make planned downtime easier. One way to do so is with clustering; for most kinds of planned maintenance, you can fail over one cluster node and upgrade it while another node in the cluster takes on its work. When the upgrade is finished, a simple failback and you're done—all (at least in theory) without drawing attention to your work by disrupting what users are doing. Another way to get much the same functionality is to use a failover solution with replication to move work over to a remote set of computers, then upgrade the local set, then fail back.

Regardless of which alternative you adopt, you should consider how much planned downtime you expect to have. Microsoft releases security patches on a regular monthly schedule, and there is usually enough advance notice of Windows and Exchange service packs for you to plan for their installation several months in advance. However, as you probably remember, these are only two of the many reasons for planned downtime—there are also hardware repairs and upgrades, scheduled interruptions in your building's electrical service, and so on. You should plan on doing maintenance at least monthly, even though there may be times when you don't actually have anything to do. What is important is that you consider how often you need to have downtime allocated, then figure out the most cost-effective way to let you keep your users in business during those periods.

Summary

This chapter builds on the others you've read so far; you learned how to apply the building blocks described in the first chapters to the design considerations you face when building an Exchange system that has both good disaster recovery capability and higher availability. The next chapter will begin exploring which buttons to push in Exchange to implement what you've learned so far.

Content Central

[Content Central](#) is your complete source for IT learning. Whether you need the most current information for managing your Windows enterprise, implementing security measures on your network, learning about new development tools for Windows and Linux, or deploying new enterprise software solutions, [Content Central](#) offers the latest instruction on the topics that are most important to the IT professional. Browse our extensive collection of eBooks and video guides and start building your own personal IT library today!

Download Additional eBooks!

If you found this eBook to be informative, then please visit Content Central and download other eBooks on this topic. If you are not already a registered user of Content Central, please take a moment to register in order to gain free access to other great IT eBooks and video guides. Please visit: <http://www.realtimepublishers.com/contentcentral/>.