

Disaster Recovery with Dfs

Douglas Toombs

(Reprinted from WindowsItPro Magazine)

When it comes to disaster-recovery planning, what constitutes a disaster? The answer might seem obvious, but you'd probably have a hard time identifying everything that you could construe a disaster. Natural disasters such as fires, earthquakes, and floods are easy to identify, but a disaster can also result from unusual circumstances. For example, I live in the Washington, DC area, and I've met dozens of individuals whose business productivity during the past 2 years has been affected because of quarantines and evacuations related to the threat of terrorist attacks, biological warfare, or criminal activity. Clearly, these types of circumstances are situations that most IT administrators might overlook when planning for a disaster.

After talking with people who've faced these situations, I developed my own definition of a disaster: Any event that keeps business users away from business data for an unacceptable length of time (sometimes permanently), thereby putting the viability of the organization at risk. I'll admit that my definition can be overly broad—even a simple situation such as a failed hard disk might meet those criteria. However, depending on the quantity and the type of data affected and the probability of being able to recover that information in a timely fashion (if at all), some organizations might easily consider a hard disk failure to be a disaster. To help you better design solutions for disaster recovery, you should begin thinking about disaster-recovery planning as a means to minimize the amount of time that your users are separated from business data, regardless of the circumstances. Let's look at a sample disaster scenario to see how Microsoft Dfs can play a part in your disaster-recovery plan and help you minimize the time you spend reconnecting users to certain types of data.

Preparing for Disaster

Imagine that your chief information officer (CIO) has asked you to develop a disaster-recovery plan (or, its newer, hipper cousin, the business continuity plan) for your organization to use in the event of a catastrophe. Your company has offices in New York and London, and the IT infrastructure between those locations consists of servers, desktops, routers, and switches. This infrastructure provides the foundation for several applications in your organization, including collaborative applications (e.g., email), vertical line of business (LOB) applications, enterprise resource planning (ERP) applications, database applications, and file storage. Given that terrorists have targeted both locations during the past decade, your CIO has asked you to plan for the worst—the loss of one of the offices for an indefinite time period.

A comprehensive disaster-recovery plan clearly will need to cover many areas to ensure business continuity through such a catastrophe. One of the most important items to consider is making the data from the impaired office available to staff members in the remaining office as soon as possible. Handing off this information lets the staff members at the remaining office carry the load and keep the business running while new facilities for the affected office are brought online.

The types of data in use in the organization typically consist of documents and files, database data, and custom-application data stores. Through the use of Dfs, you can make the first data category (i.e., documents and files) highly available during a disaster. With a bit of careful planning and minimal ongoing effort, Windows 2000 can store your data in multiple locations and make sure it's always up-to-date.

A Dfs Primer

Dfs is a core part of Win2K, but the name is misleading. Dfs isn't a true file system like NTFS or FAT, nor is it a means of organizing the way you store data on a magnetic or optical storage device. Rather, Dfs is a mechanism for organizing the way you present shared data to users, and (optionally) keeping replicated copies in multiple locations.

Disaster Recovery with Dfs

Douglas Toombs

(Reprinted from WindowsItPro Magazine)

In a nutshell, Dfs lets you build virtual file shares on your network and point those shares to physical file shares, possibly across multiple servers. End users can connect to these virtual file shares as if they were connecting to physical file shares. When users connect to these virtual file shares, Dfs directs them to a copy of the data that's available in their location or to any one of several replicated copies available on the network.

The plumbing behind this magic is known as the File Replication Service (FRS). The FRS is built into Win2K and doesn't require any setup (Dfs is the only component that I discuss that you must configure). Let's look at a sample Dfs configuration suitable for disaster-recovery purposes.

Planning for Dfs

Imagine that you have several file servers in both the London and New York offices, and you've decided that you want to replicate all file-sharing data (e.g., documents, spreadsheets, Microsoft PowerPoint presentations, marketing materials) from one office to the other. This data is spread across multiple servers at each site, and you'd like to back up the London office data to one server at the New York office and vice versa.

To start, you need to consider the amount of disk space you'll need. If you have approximately 50GB of documents and files in each office, you need to make sure that you have an additional 50GB of disk space available in each office to store the replicated data.

Next, consider the bandwidth needed to support replication between sites. For example, if you have a T1 circuit between your office locations, you have approximately 1.5Mbps of bandwidth to support all data communications from one site to the other. You'll need to determine how much of that bandwidth is being used by existing site-to-site communication—your WAN administrators should be able to provide this information. If this information isn't readily available, you can consider using an SNMP-collection tool such as the Multi Router Traffic Grapher (MRTG) to track bandwidth usage over time. Whichever means you use, subtract the amount of bandwidth in use from the total amount available. For the purposes of our example scenario, let's assume that half the bandwidth between the New York and London offices is already in use, which leaves 750Kbps of bandwidth.

The final data element to consider is how much end-user data changes at each office location on a daily basis. Dfs replicates changes down to the file level, so every time someone modifies a 20MB PowerPoint presentation on a replicated share, Dfs will push the entire 20MB file around your WAN as a part of the replication process. Obviously, replicating data in this manner can introduce large amounts of traffic on your network.

You can take several approaches to determine how much data changes on a daily basis in your organization—for example, I use Windows' advanced Search feature to locate all files that have changed within the past day. Run this search in the directories you want to replicate, then add up the total size of all the files you find. Some files will be updated several times a day, so you might need to increase your estimate to factor in multiple replications for some of these files.

After you estimate how much data (in bytes) users modify in a typical day, multiply that amount by 8 to determine the number of bits that it represents. For example, imagine that our example organization modifies 200MB of files on an average day. If you multiply 200MB by 8, you'll need to replicate 1.6 billion bits of data. Divide this value by the total amount of bandwidth available to calculate the number of seconds worth of Dfs replication traffic on your network in a given day (in the case of our example, this value is 2133 seconds per day). Given that there are 86,400 seconds in a day, the calculated Dfs replication traffic time probably accounts for a reasonable amount of data to move

Disaster Recovery with Dfs

Douglas Toombs

(Reprinted from WindowsItPro Magazine)

across a T1 connection, even if several sites are all being updated at the same time (FRS replication is multi-master and can process several sites simultaneously).

You can get fancier with this calculation process by using a window of less than 24 hours to determine how much data changes in a day and how many seconds are available in that time frame. (Also, keep in mind that we're not taking into consideration the packet-header overhead, which can add a significant amount of additional bits, typically 56 bytes per packet.) Because the Dfs replication traffic between the two locations will have the most noticeable effect during core business hours, you can expect that the business day between the New York and London offices represents 13 hours out of the day (assuming a typical 8-hour day, plus a 5-hour time difference). After you determine that you have the disk space and the bandwidth to start Dfs data replication, it's time to start planning and implementing.

Designing for Dfs

In a standard Win2K setup without Dfs, the OS typically stores user files in directories on a server and shares those directories with the end users. To access files within these shared directories, the end users map or connect to these network shares. The process of connecting to shares makes finding data on the network easy for end users and consists of four functional layers starting from the end user with the share layer, the folder layer, the file layer, and the physical disk layer. Depending on how you plan to design your Dfs infrastructure, you might need to add another layer or two—the mandatory Dfs root layer, as Figure 1 shows, and the optional Dfs link layer, as Figure 2 shows.

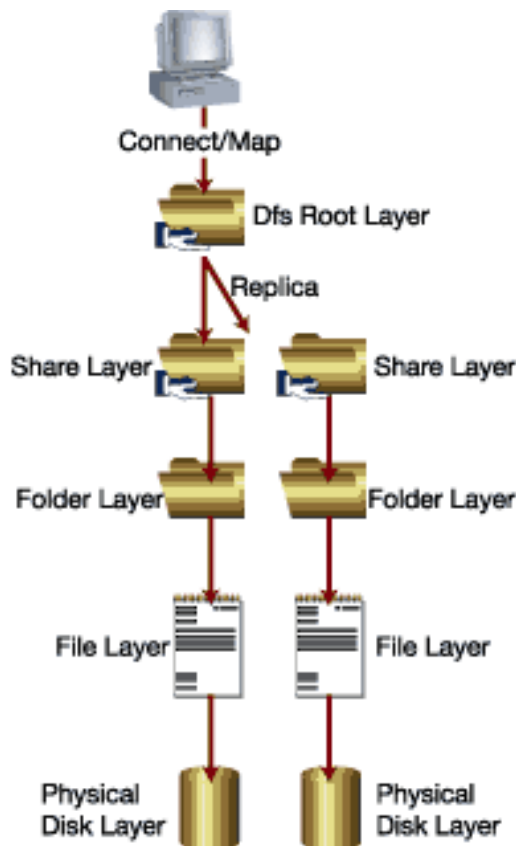


Figure 1: Establishing a Dfs root layer replica share

Disaster Recovery with Dfs

Douglas Toombs

(Reprinted from WindowsItPro Magazine)

The Dfs root layer is similar to the root of a hard disk—every disk must have one, and a Dfs structure is no exception. When you build your Dfs structure, you need to define one or more shares on your network to act as the Dfs root. For redundancy purposes, you'll want to define root shares on several servers within your Dfs structure. Let's look at the process of implementing a Dfs root structure using a test lab in the Active Directory (AD) domain netarchitect local, with data replication occurring between two servers: ARCHITECT10 and DMZSERVER.

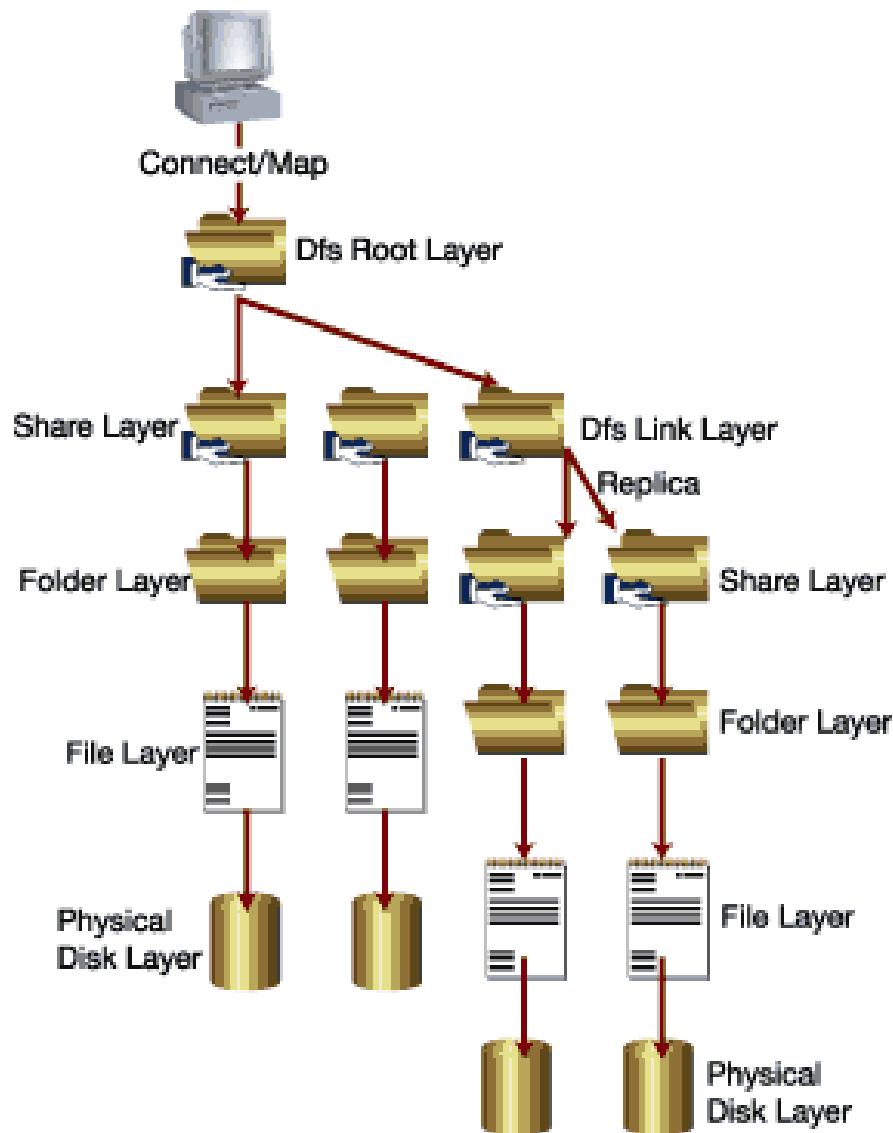


Figure 2: Establishing a Dfs link layer replica share

Configuring the Dfs Root

In Win2K, you start the process of defining a Dfs root in your organization by launching the Microsoft Management Console (MMC) Distributed File System snap-in under Start, Programs, Administrative Tools. Right-click Distributed File System in the snap-in's left-hand pane, then select New Dfs Root to

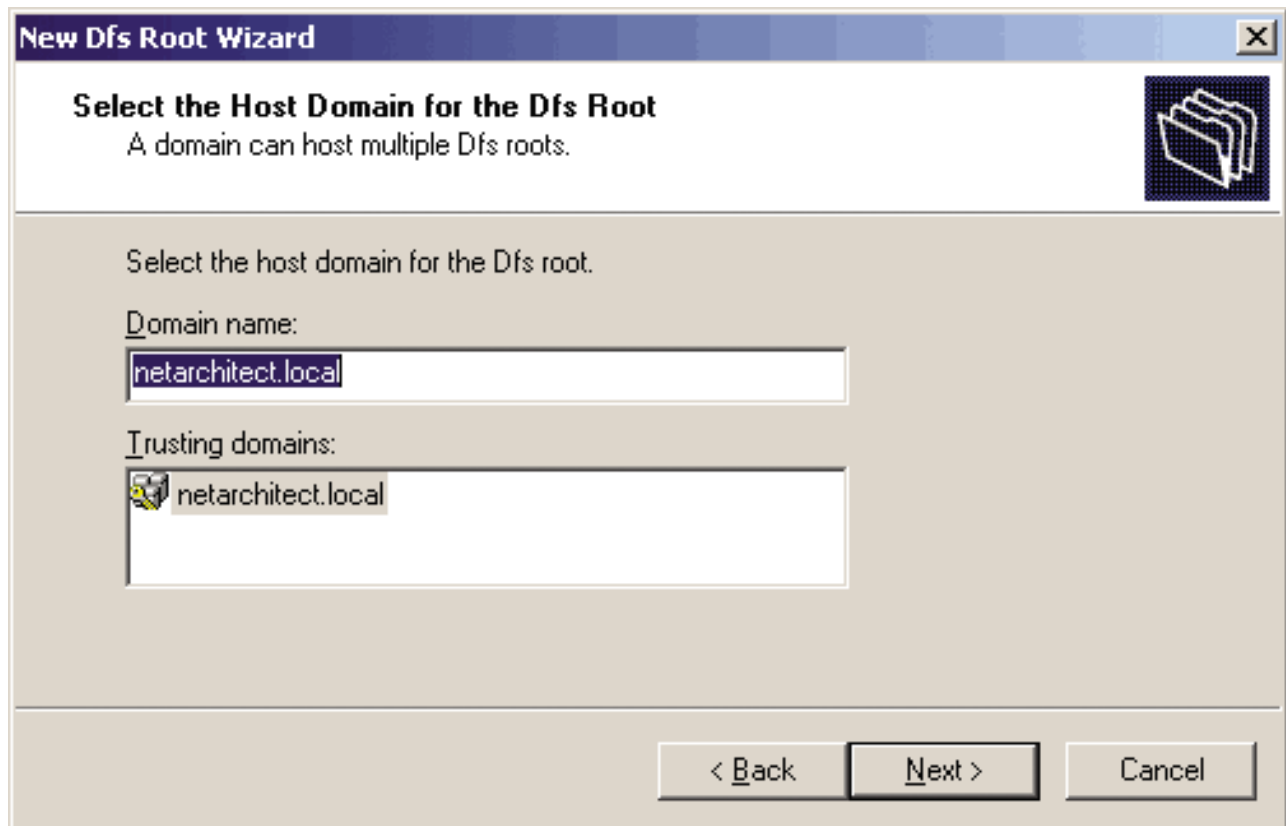
Disaster Recovery with Dfs

Douglas Toombs

(Reprinted from WindowsItPro Magazine)

launch the configuration wizard. To begin, you need to decide whether to create a domain Dfs root or a standalone Dfs root. Because you're defining Dfs to support file replication between sites, a domain Dfs root is the only acceptable choice.

Next, you need to decide which AD domain you want to use to host the Dfs root that you're creating. Choose the domain that's appropriate for your situation, as Figure 3 shows. The domain name you select at this point will be the name that end users use to connect to your Dfs root structure. For example, when users want to connect to the sample Dfs root, they'll map to \\netarchitect.local\Dfs root name, instead of mapping directly to a machine.



After you select the domain, you need to define the host server for the Dfs root, then you need to identify which share you want to use. If you didn't create a share before you started this process, you can create one on the fly. The share you use must point to a directory on an NTFS drive because FRS uses NTFS 5.0's (NTFS5's) journaling system to monitor when a file has changed. For the purposes of our example, I've created a share called netarchitect-dfsroot on the ARCHITECT10 server. You can use any name you like, but I find that using descriptive share names helps me keep things straight when working with many folders, shares, and replicas.

The final step in configuring the Dfs root is to assign it a name, as Figure 4 shows. As with the domain name, users will use this Dfs root name to connect to the Dfs structure. For this reason, I prefer to use a name that describes the function of the share (e.g., ReplicatedFiles, SyncFiles).

Disaster Recovery with Dfs

Douglas Toombs

(Reprinted from WindowsItPro Magazine)

New Dfs Root Wizard

Name the Dfs Root
You must provide a unique name for each Dfs root.

Type a name for the Dfs root.
\\netarchitect.local\SyncFiles

Dfs root name: SyncFiles

Comment: Enterprise Dfs Root

< Back Next > Cancel

After you complete this step, you should be able to test your new Dfs root from a workstation by connecting to the \\domainname\Dfsrootname Universal Naming Convention (UNC) name that you defined in the Dfs root creation process (e.g., \\netarchitect.local\SyncFiles). Your workstation will retrieve the Dfs definition data from AD, then find the underlying share or shares for the root.

Adding Redundancy to Dfs

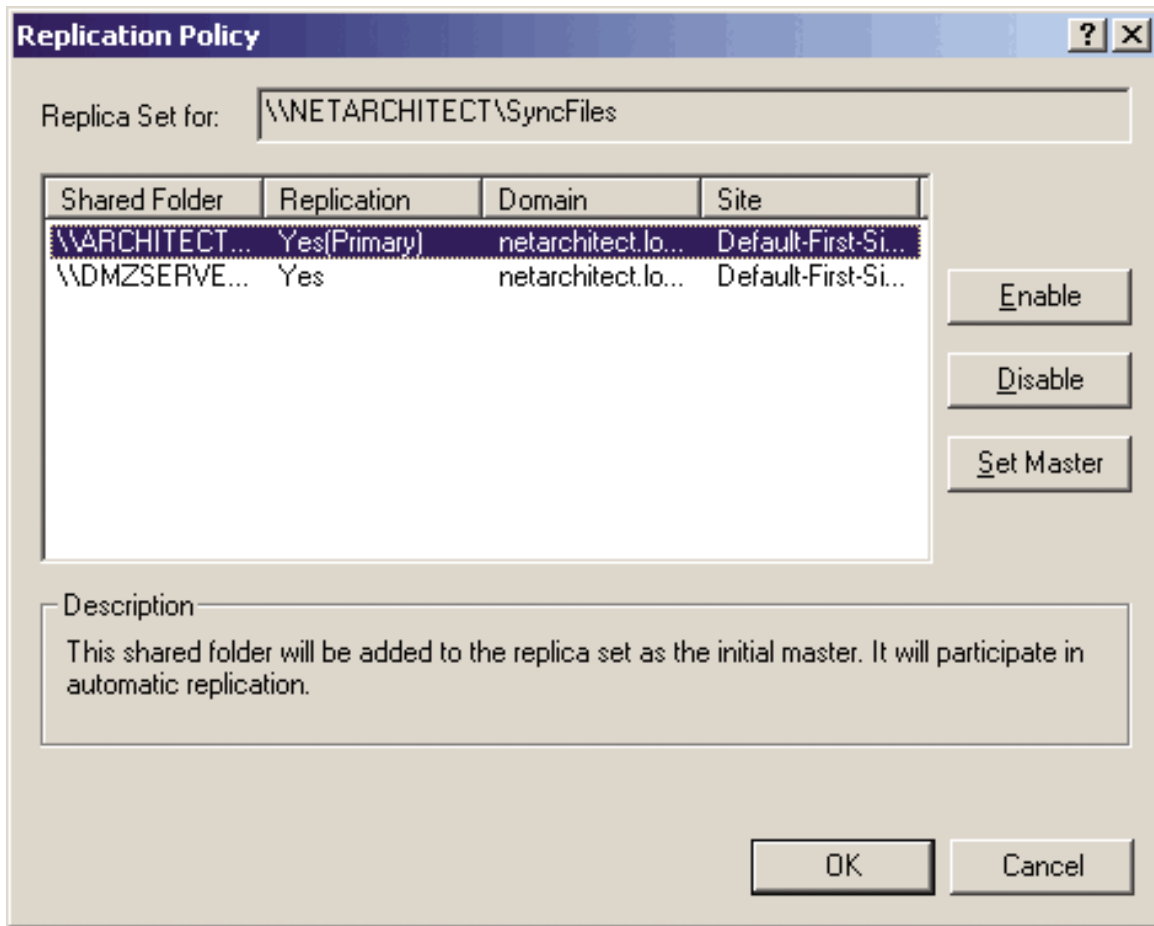
At this point, you've created a root for your Dfs structure, but the root isn't redundant. So, the next step is to define a Dfs root replica by right-clicking Distributed File System in the left-hand pane of the Distributed File System snap-in. From the context menu, select New Root Replica to start selected portions of the Dfs Root Wizard. First, you need to select the machine that you want to host the Dfs root replica, which in the case of our example is DMZSERVER. Second, you must select the appropriate share that you defined before you started the wizard, which in the case of our example is netarchitect-dfsroot.

After you define the share for the Dfs root on your second server, you need to configure replication between both shares. By default, Dfs doesn't automatically set up data replication for you, so you must enable it. Right-click the Dfs root you've created, then select Replication Policy from the context menu to open the Replication Policy dialog box, which Figure 5 shows. Click Enable to enable replication for each of the shares, then decide which share should be the master for the first-time replication. Select Set Master for that share. After you complete these steps, Win2K will replicate the data on these shares, then keep them synchronized. After the initial replication has occurred, the distinction between the Primary share and the other shares disappears.

Disaster Recovery with Dfs

Douglas Toombs

(Reprinted from WindowsItPro Magazine)



Depending on how you intend to implement Dfs, you might be finished setting up Dfs replication for your environment. At this point, you can start putting folders, subfolders, and files into your Dfs root structure and Win2K will automatically keep the data synchronized between all the shares in the Dfs root. More specifically, you can lay out an entire subdirectory structure so that all crucial file-based data resides somewhere within your Dfs structure. Then, you'll be able to ensure that in the event of a disaster, this information will be replicated and available at other locations within your organization.

If you want to go further with Dfs, you can create Dfs links within your Dfs structure. Dfs links appear to end users as folders within the Dfs structure, but they're actually new shares defined within the Dfs structure (separate from the shares you used to define the Dfs root), as Figure 2 shows. Dfs links are especially useful when you have various types of data that already exists across multiple servers in your office and you want to consolidate that information into one easily navigable structure of shares and directories. To define a Dfs link, right-click a newly defined Dfs root in the left-hand pane of the Distributed File System snap-in, then select New Dfs Link to start a configuration wizard. Much like the Dfs root configuration, the wizard will ask you to select the servers and shares that will participate in the link and whether you want to replicate between shares.

Dfs Tips, Tricks, and Caveats

Before you get too excited about Dfs, realize that it has some limitations—it isn't the final solution for all your offsite data-replication needs, but it's a useful tool to get you started. The first caveat is to recognize what types of data you can replicate with Dfs. For example, Dfs works well with files that users open, change, then close until they need to access those files again. Dfs recognizes that the file

Disaster Recovery with Dfs

Douglas Toombs

(Reprinted from WindowsItPro Magazine)

has changed and replicates the new version of the file across your network, as needed. Again, Dfs replicates the entire file, not just the changes. Therefore, if you update a large file, Win2K will copy the entire new file to each of the replicas in your network.

In addition to moving data files from one replica to another, Win2K also replicates permissions between the replicas that you define in the Dfs structure. If you add or remove permissions to a folder or file in one location, Dfs will replicate those changes throughout the environment.

The only items that Dfs won't replicate are file locks—the indicators that Win2K uses to determine whether someone else is working on a file. This consideration is important and will most likely affect how you choose to define your Dfs structures. Because Dfs doesn't replicate file locks, two users could be working on the same file at the same time, each with a different copy of the file and completely unaware of the other copy. If this situation takes place and both users save their changes to the network at roughly the same time, a last-writer-wins methodology occurs. For example, one user's changes might be written to disk and synchronized, only to be immediately overwritten by changes saved by the other user. To help avoid this situation, I typically advise people to always consider synchronized replicas as backup shares only, ones that users should never access directly.

One way I typically synchronize replicas as backup shares is to define shares according to the individual sites that own documents, then define a backup share in another site. For example, a repository of sales documents from the New York office might be in a share called NYSALESDOCS. To replicate this information to London, I would create a directory, share it as NYSALESDOCS-BACKUP, then define it as a Dfs replica. This naming convention helps clarify what the share on the London server is all about.

As you can probably guess, if only one master SALESDOCS Dfs share exists for the global organization, Dfs will have a hard time forcing users into a specific share on a specific server because Dfs uses AD sites to determine whether a share is nearby.

Because Dfs replication is triggered by a file change, Dfs doesn't work well with database files that are always open—database files never really encounter a "close" operation, so the data is never synchronized. As a result, Dfs won't help you replicate your Microsoft SQL Server or Microsoft Exchange Server databases.

Keep in mind that Dfs replication is similar to disk mirroring (i.e., the system replicates whatever data it's given, good or bad). As a result, if a virus infects a user's workstation, corrupts all *.doc files, and users modify those files, Dfs will replicate those corrupted files as long as they're in the Dfs structure. This scenario is one example of why traditional backups are still necessary. Finally, by default Dfs can't replicate certain files: .bak and .tmp files and any filename beginning with a tilde (~).

Only a Partial Solution

Dfs is an affordable and reliable mechanism for providing disaster-recovery capabilities for certain types of data in your organization. Considering you've already paid for the capability, you can implement Dfs as part of your disaster-recovery plan and use it to the best of its abilities—with the sincere hope that you'll never end up depending on it.