



Operating System

Distributed File System: A Logical View of Physical Storage

White Paper

Abstract

The Microsoft® Distributed File System (Dfs) is a network server component that makes it easier for you to find and manage data on your network. Dfs is a means for uniting files on different computers into a single name space. Dfs makes it easy to build a single, hierarchical view of multiple file servers and file server shares on your network.

Microsoft Distributed File System version 4.1 for Microsoft Windows NT® Server 4.0 is currently available for download from the Microsoft Web site at <http://www.microsoft.com/ntserver>. This release includes the Microsoft Windows® 95 operating system Dfs client and enhanced security signatures. In addition, Microsoft Windows® 2000 will include directory service-enabled enhancements to Dfs. This paper covers Dfs technology as a whole, including the version for Windows 2000.

© 1999 Microsoft Corporation. All rights reserved.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Microsoft, Active Desktop, BackOffice, the BackOffice logo, MSN, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

*Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA
0399*

CONTENTS

INTRODUCTION	1
What is a Distributed File System?	1
Benefits of Dfs	2
DFS OVERVIEW	4
Flexible Administration	4
Easy Browsing of File Servers	4
Simple Searches for Files or Data	5
Better Data Availability and Load Balancing	5
Name and Location Transparency	5
Flexible Internet and Intranet Management	6
Integration with Windows 95, Windows 98, and Windows NT Workstation Clients	7
Simplified File Maintenance Tasks	7
TECHNICAL OVERVIEW OF DFS	9
Dfs Concepts, Terms, and Conventions	9
Dfs Root	9
Accessing a Dfs Volume	9
Hosting a Dfs Volume	9
Post-Junction Junctions	11
Alternate Volumes	11
Down-Level Volumes	11
Partition Knowledge Table	12
Dfs Referrals: How Junctions Are Resolved from Logical Names into Physical Addresses	12
Fail-over Between Alternates Volumes	13
Scenario #1	14
Scenario #2	14
Scenario #3	14
Scenario #4	14
Security	14
Session Setups	14
ACLs	15
Considerations for Developers Writing Applications that Use Dfs	16
Free Space	16
Connections	16
File Systems	17
Backup and Restoration	17
TAKING ADVANTAGE OF DIRECTORY SERVICES	18
APPENDIX I: DFS PROGRAMMING INTERFACES	19
Public Interfaces	19
NetDfsAdd	19
Parameters	19

Return Values	20	
NetDfsRemove	20	
Parameters	20	
Return Values	20	
NetDfsEnum	20	
Parameters	21	
Remarks	22	
NetDfsGetInfo	22	
Parameters	22	
Return Values	23	
Remarks	23	
NetDfsSetInfo	23	
Parameters	24	
Return Values	24	
Remarks	24	
Dfs API Error Values	25	
APPENDIX 2: CIFS SPECIFICATION.....	26	
Dfs Support	26	
Dfs Path Names	26	
TRANS2_GET_DFS_REFERRAL: Retrieve Distributed File-system Referral	27	
TRANS2_REPORT_DFS_INCONSISTENCY: Inform a Server About Dfs Error	30	
APPENDIX 3: DIRECTORY SERVICE OBJECTS	32	
Dfs-Configuration	32	
FT-Dfs	32	
PKT	32	
PKT-Guid	33	
Remote-Server-Name	33	
FOR MORE INFORMATION.....	34	

INTRODUCTION

The Distributed File System (Dfs) for the Microsoft® Windows NT® Server and Microsoft Windows® 2000 Server operating systems is a network server component that makes it easier for you to find and manage data on your network. Dfs is a means for uniting files on different computers into a single name space. Dfs makes it easy to build a single, hierarchical view of multiple file servers and file server shares on your network. Instead of seeing a physical network of dozens of file servers, each with a separate directory structure, users will now see a few logical directories that include all of the important file servers and file server shares. Each share appears in the most logical place in the directory, no matter what server it is actually on.

Dfs does for servers and shares what file systems do for hard disks. File systems provide uniform named access to collections of sectors on disks; Dfs provides a uniform naming convention and mapping for collections of servers, shares, and files. Thus, Dfs makes it possible to organize file servers and their shares into a logical hierarchy, making it considerably easier for a large corporation to manage and use its information resources. In addition, Dfs is not limited to a single file protocol and can support the mapping of servers, shares, and files, regardless of the file client being used, provided that the client supports the native server and share.

What is a Distributed File System?

Dfs provides name transparency to disparate server volumes and shares. Through Dfs, an administrator can build a single hierarchical file system whose contents are distributed throughout your organization's WAN. In short, Dfs can be thought of as a share of other shares.

Historically, with the universal naming convention (UNC), a user or application was required to specify the physical server and share in order to access file information (that is, the user or application had to specify `\\Server\Share\Path\Filename`). Even though UNC's can be used directly, a UNC is typically mapped to a drive letter where `x:` might be mapped to `\\Server\Share`. From that point, a user had to navigate beyond the redirected drive mapping to the data he or she wishes to access (for example, copy `x:\Path\More_path\....\Filename`).

As networks continue to grow in size and as organizations begin to use existing storage—both internally and externally—for purposes such as intranets, mapping a single drive letter to individual shares scales poorly. Further, although users can use UNC names directly, these users can be overwhelmed by the number of places where data can be stored.

Dfs solves these problems by permitting the linking of servers and shares into a simpler, more meaningful name space. This new Dfs volume permits shares to be hierarchically connected to other Windows shares. Since Dfs maps the physical storage into a logical representation, the net benefit is that the physical location of data becomes transparent to users and applications.

Benefits of Dfs

The features of Dfs and their benefits are outlined in detail in the following table.

Feature	Description	Benefits
Custom hierarchical view of shared network resources	By linking shares together, administrators can create a single hierarchical volume that behaves as though it were one giant hard drive. Individual users can create their own Dfs volumes, which in turn can be incorporated by other Dfs volumes. These are called inter-Dfs links.	Provides a simplified view of network shares that can be customized by the administrator.
Flexible volume administration	Individual shares participating in the Dfs volume can be taken offline without affecting the remaining portion of the volume name space.	Allows administrators to manage physical network shares, independent of their logical representation to users.
Graphical administration tool	Each Dfs root is administered with an easy-to-use graphical administration tool that permits browsing, configuration of volumes, alternates, and inter-Dfs links, as well as administration of remote Dfs roots.	Requires little training, reducing the need for trained, full-time server administrators.
Higher data availability	Multiple copies of read-only shares can be mounted under the same logical Dfs name to provide alternate locations for accessing data. If one of the copies becomes unavailable, an alternate is automatically selected.	Important business data is always available, even if a server, disk drive, or file occasionally fails.
Load balancing	Multiple copies of read-only shares on separate disk drives or servers can be mounted under the same logical Dfs name, thus permitting limited load balancing between drives or servers. As users request files from the Dfs volume, they are transparently referred to one of the network shares comprising the Dfs volume.	Automatically distributes file access across multiple disk drives or servers to balance loads and improve response time during peak usage periods.
Name transparency	End users navigate the logical name space without consideration to the physical locations of their data. Physical data can be relocated to any server and the logical Dfs name space can be reconfigured so that the end user's perspective of the Dfs name space is unaffected (that is, it is transparent to users that their data has changed location).	Increased administrative flexibility. Administrators can move network shares between servers or disk drives without affecting users' ability to access the data.
Integration with Windows NT security model	No additional administrative or security issues. Any user who connects to a Dfs volume is only permitted to access files for which he or she has appropriate rights on that share.	Uses the existing Windows NT security model for easy administration and secure access.
Dfs client integrated into Windows NT Workstation 4.0, available for Windows 95 and Windows 98	The Dfs Windows NT Workstation client has been incorporated into Windows NT Workstation 4.0. This integration with the SMB redirector allows the extra Dfs features to be fully pageable and does not affect memory needs or standard client access performance.	Dfs functionality requires no additional resources on client systems.

Feature	Description	Benefits
Intelligent client caching	A Dfs volume can potentially connect hundreds or thousands of published shares. The client software makes no assumptions about what portion of Dfs published information a user might access. As a result, the first access of a published directory caches certain information locally. The next time a client accesses that portion of the Dfs name space, the cached referral is accessed, rather than obtaining a new referral.	Allows high-performance access to complex hierarchies of network volumes.
Windows 95 and Windows 98 Client	In addition to the Dfs-aware Windows NT Workstation redirector that ships with Windows NT Workstation 4.0, Dfs includes a service to permit Windows 95 and Windows 98 users to navigate the Dfs name space. With the current release of Dfs, Windows clients can only access non-SMB volumes through a server-based gateway (for example, Microsoft Gateway Services for NetWare, which is included with Windows NT Server).	Extends Dfs benefits to Windows 95 and Windows 98 users.
Interoperates with other network file systems	Any volume that is accessible through a redirector on Windows NT Workstation can participate in the Dfs name space. This can be through either client redirectors or server-based gateway technology.	Administrators can create a single hierarchy incorporating heterogeneous network file systems.

DFS OVERVIEW

Flexible Administration

Dfs provides a useful tool for reducing the complexity of managing an ever-changing server configuration. As old servers are retired, groups of servers are consolidated into more powerful servers, and new servers are brought online for the ever-growing storage requirements. In principle, administrators should be required to manage fewer and fewer servers and allowed to more easily keep up with new technologies as they become available.

Dfs allows administrators to build hierarchical directories that span multiple file servers and file shares. Dfs includes an easy-to-use administration tool that makes it simple for network managers to build and maintain Dfs directories. Dfs also includes a scriptable command line administration tool. Dfs is easy to manage because no new tools are required for most management tasks. File security is managed using the Windows NT Server security model and Windows Explorer.

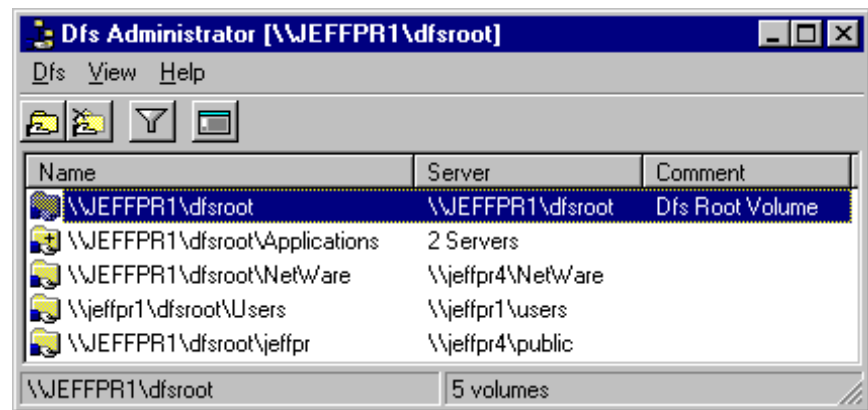


Figure 1: Dfs Administrator, managing a Dfs volume

Dfs makes it easy for network managers to replace servers. Each node in the Dfs directory tree is assigned a logical name that points to a file share. The Dfs node can be switched to point to a new server while the old server is taken offline. Users never know that they are using a different server, since the Dfs directory tree does not change.

Easy Browsing of File Servers

Dfs makes it easy to create a single directory tree that includes multiple file servers and file shares in a group, division, or enterprise. Dfs gives the user a single directory that can span a vast number of file servers and file shares, making it easy to browse the network to find the data and files needed. Browsing the Dfs directory is easy because Dfs sub-directories can be assigned logical, descriptive names—no matter what the name of the actual file server or file share is.

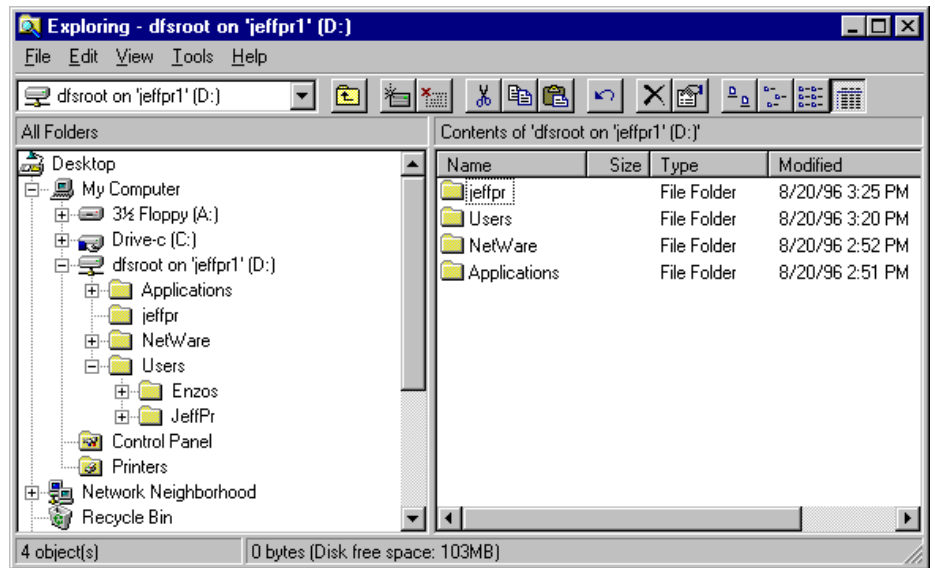


Figure 2: Browsing the Dfs volume shown in Figure 1

Dfs directories make it easy for a group to organize all the relevant file shares into a single Dfs Directory tree. Server-level Dfs directories can be combined into a hierarchical Dfs directory, constrained only by the limit of 260 characters per file path.

Simple Searches for Files or Data

Dfs makes it easy to find data and files because the file search tools included in Windows 95, Windows 98, and Windows NT Workstation, or even a word processing program, can search for a specific file that could be located on any server in the Dfs directory tree.

With Windows 95, Windows 98, and Windows NT Workstation, Dfs makes large, distributed networks easier to use. Instead of having to deal with multiple persistent network connections to separate physical file servers, each user needs only one or more persistent network connections to their Dfs trees.

Better Data Availability and Load Balancing

Dfs can also provide enhanced data availability by pointing to multiple volumes that can be alternates for each other. If a volume is unavailable, Dfs hands the request to the alternate one. In addition to enhanced availability, Dfs provides load balancing performance gains because it can evenly distribute client access to Dfs volumes across multiple alternate network shares. If 300 users require access to one volume, Dfs can split the users among copies on two or more servers to balance the load.

Name and Location Transparency

Dfs transparently links server volumes and shares into a single name space,

providing name transparency. This helps administrators by simplifying growth and administration of the network through centralized management.

Name transparency eliminates the need for end users to know where information is physically stored. This is a particular advantage in large organizations, which may have many areas where information is stored. Dfs allows a user to find a particular piece of information within a meaningful hierarchy without having to know where (or on which computer) it is stored.

The location transparency of Dfs eases the burden of upgrading to new computers by allowing additional storage to be published in subdirectories. Because users do not need to know the name of a server or share, administrators can physically move user information to another server without having to re-educate users (or reconfigure applications and shortcuts) about how they can find their data. Additional or higher-performance servers can be added without changing their name. Dfs allows one share to span multiple servers; thus, new computers could be presented as new directories within the share.

Flexible Internet and Intranet Management

Dfs simplifies deployment of Internet and intranet solutions. A Web master can now build a logical Dfs directory that includes the default Web pages of each department's Web server as a subdirectory of the main Internet or intranet Web. This allows each department or group to retain control over their unique intranet content and applications, while the user only sees a single, unified Internet site or intranet.

Dfs works with the built-in Internet Information Server of Windows NT Server. The HTML links to other pages stored in Dfs do not have to be updated if the initial page is physically moved from one server to another, provided an administrator reconfigures Dfs accordingly. This allows a user with an Internet page to publish it: if the server hosting that page is removed and the page is republished in a different location, all links pointing to that page do not have to be reconfigured.

Integration with Windows 95, Windows 98, and Windows NT Workstation Clients

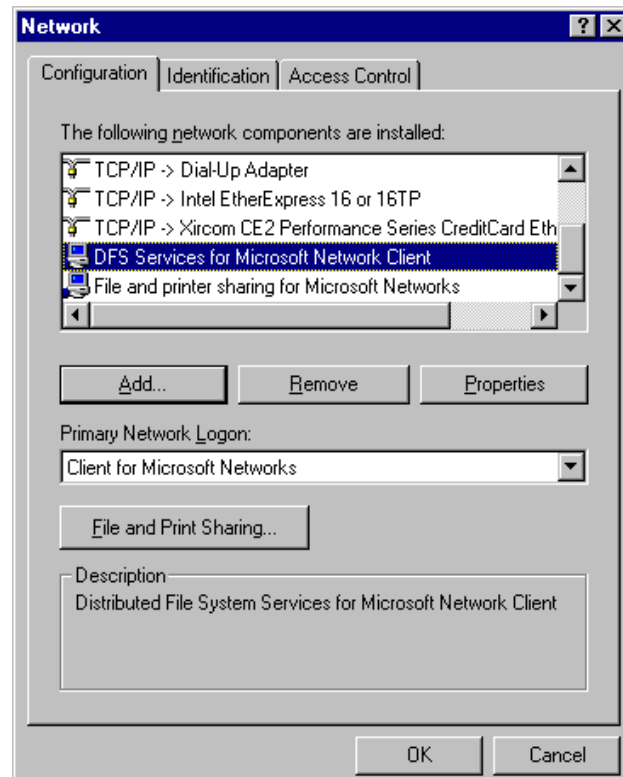


Figure 3: Installing the Dfs service for Windows

Dfs support is already included in Windows NT Workstation 4.0, and support for Windows 95 and Windows 98 is included in the Dfs release. Dfs can make Windows 95, Windows 98, and Windows NT Workstation desktops easier to use by eliminating the need for multiple persistent network connections to individual physical servers. Also, workstation shares may participate in the server-based Dfs volume, making it easy to share desktop files with others and helping network administrators extend corporate file maintenance down to desktops. Additionally, with the Windows NT Client for Dfs, you can Net Use below the \\Server\Share level.

Simplified File Maintenance Tasks

Dfs makes file maintenance tasks such as enterprise backups easy. Since a single Dfs tree can be built to cover a large number of servers, the backup software can back up this single tree, no matter how many servers or shares are part of the Dfs. Dfs can also make backups of data on end-user systems easy. Because Windows 95, Windows 98, and Windows NT Workstation systems can be "leaves" in a Dfs tree, a Dfs tree can include all the directories on users' desktops that network managers want backed up. Other file maintenance tasks that Dfs can help simplify

include scanning for viruses across an entire network and content indexing of data for use with search tools, such as Microsoft Index Server.

Care must be taken to have the Dfs tree in place before restoring, as Dfs is transparent to almost all backup utilities available today.

TECHNICAL OVERVIEW OF DFS

Dfs Concepts, Terms, and Conventions

Dfs Root

A Dfs Root is a local share that serves as the starting point and host to other shares. Any shared resource can be published into the Dfs name space.

Accessing a Dfs Volume

A Dfs volume is accessed using a standard UNC name:

```
\\Server_Name\Dfs_Share_Name\Path\File
```

where *Server_Name* is the name of the host Dfs computer name, *Dfs_Share_Name* maps to any share that is designated to be the root of your Dfs, and *Path\File* is any valid Win32® path name. You can *not* use a drive to any place in the Dfs. Dfs provides a level of indirection between such a UNC name and the underlying server and share that is actually hosting the file or directory.

Shares participating in a Dfs may be hosted by any server accessible by clients of the Dfs. This includes shares from the local host, shares on any Windows NT server or workstation, or shares accessible to Windows NT through client software (NetWare, Banyan, and so on).

With the directory service-enabled version of Dfs, a Dfs volume can also be accessed by both its fault-tolerant name and its domain name:

```
\\Fault_Tolerant_Name\Dfs_Share_Name\Path\File  
\\Domain_Name\Fault_Tolerant_Name\Path\File
```

where *Fault_Tolerant_Name* is a name that the administrator selects to specify a Dfs volume stored in the DS (this may be a group of computers providing root level fault tolerance), and *Domain_Name\Volume* is the name of a standard volume object in Windows directory services.

Hosting a Dfs Volume

A network can host many individual Dfs volumes, each having a distinct name. Any Windows NT 4.0 or Windows 2000 server can run the Dfs service and host a Dfs volume. For the Windows NT 4.0 release of Dfs, a server can host one and only one Dfs. There are no architectural limits that prevent a Windows NT-based operating system from hosting multiple Dfs volumes in future releases.

A typical Dfs scenario would have a Dfs volume composed of link volumes from multiple servers throughout an organization. For example, consider an organization in the banking industry that has a pool of workers who handle word processing needs. This pool of workers would need access to documents stored in every department and branch throughout the organization. A single Dfs volume hosted locally could tie together only those servers and the specific share points that are typically used to store word processing information. The following table shows a typical Dfs.

UNC Name	Maps to	Description
\\Server\Public	\\Server\Public	Root of the organization's Dfs
\\Server\Public\Intranet	\\IIS\Root	Junction to the intranet launching point
\\Server\Public\Intranet\ CorpInfo	\\Marketing\Info\ Corporate_HTML	Junction to departmental intranet content
\\Server\Public\Users	\\Server\Public\Users	Collection of home directories
\\Server\Public\Users\ Bob	\\Server\Public\Users	Junction from Users to Bob's directory on the corporate development server
\\Server\Public\Users\ Bob\Java_Apps	\\Bob1\Data\ Java_Apps	Junction point from Bob's development directory to one of Bob's personal workstations
\\Server\Public\Users\ Bob\Java_Apps	\\Bob2\Backups\ Java_Apps	ALTERNATE Volume: Manually maintained backup of Bob's work
\\Server\Public\Users\ Ray	\\Server\Public\Users	Down-level Volume : junction to a non-SMB volume (such as NetWare or NFS)

Figure 4 below shows a hierarchical diagram of the information in the table above.

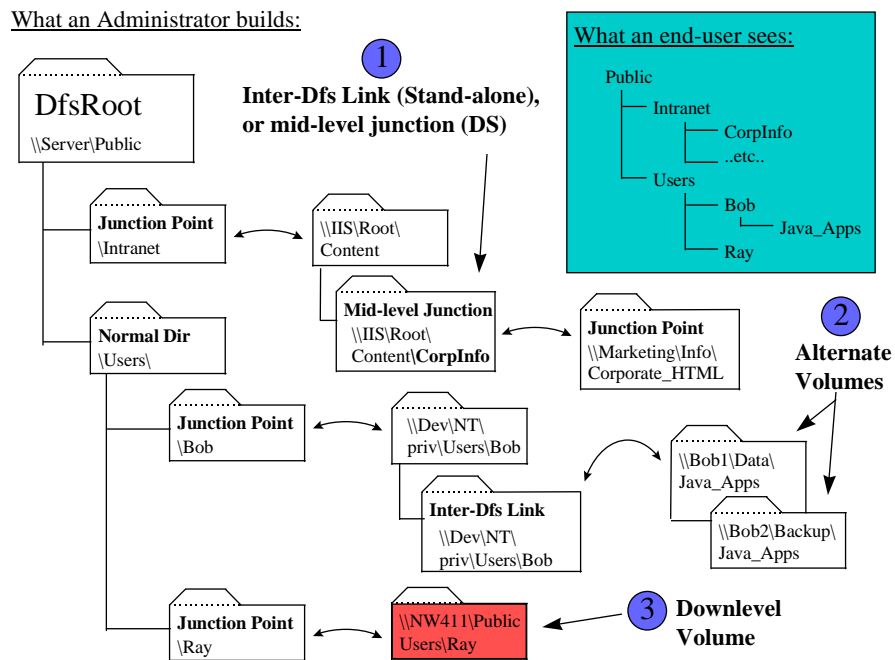


Figure 4: Typical Dfs

In addition to representing the table of junctions above, the following important examples are presented in Figure 4:

- Post-junction junctions
- Alternate volumes
- Down-level volumes

These are described in detail below.

Post-Junction Junctions

This is a junction that has child junctions. There are two methods of setting this up: inter-Dfs links and midlevel junctions.

Inter-Dfs Links (Stand-Alone Server)

It is possible to join one Dfs volume to another Dfs volume. As an example, individual departments within an organization can choose to set up and administer a logical name space that matches their work needs. The corporate Dfs volume could link together all departmental Dfs volumes. The effect of an Inter-Dfs link is that as you cross the junction, you have changed the server that you reference as your Dfs root. Although this is transparent to the end user, the Dfs client now receives its referrals from the new Dfs root.

Midlevel Junctions (Directory Service-enabled Version Only)

This is a planned feature for a future release of Dfs to support unlimited hierarchical junctioning that does not require inter-Dfs links. All referrals will be resolved from the original Dfs root, reducing points of failures and minimizing the number of referrals required to resolve deeply nested paths.

Alternate Volumes

If two (or more) shares in an organization are exact replicas of one share and you want to leverage the volumes, you can mount them at a single point in your Dfs name space.

Dfs makes no attempt to ensure that the data stored between the two shares is replicated. Dfs replicas should be considered alternative sources of synchronized or replicated data (replication can be manual or automatic). If content is not replicated, alternate volumes can still be beneficial in a read-only scenario where any data writes are performed through physical naming across all alternates.

Dfs has a limit of 32 alternates for any given junction point; there is no limit to the number of junctions created at any point in your Dfs tree.

Down-Level Volumes

Any volume that is Windows NT Server 4.0 or later can host a Dfs volume and participate as an inter-Dfs link. All other volumes are considered to be *down-level*. Down-level volumes can be published in the Dfs volume but cannot host Dfs or be joined to other volumes. Down-level volumes include Windows NT Workstation (all versions), Windows 98, Windows 95, Windows for Workgroups, and all non-Microsoft shares for which client redirectors are available. Windows 95 and Windows 98 Dfs-aware clients can take referrals for all up-level volumes and all

SMB (Microsoft) down-level volumes, but cannot negotiate non-SMB volumes.

Partition Knowledge Table

The partition knowledge table (PKT) holds knowledge of all junction points. It is administered by the Dfs Admin API shown in Appendix 1.

The PKT is sorted look-up table (~300 bytes per entry)

Dfs Path	[Server + Share] (a list)	Time to Live

Locally (client side):

One per Dfs volume mapped by the client. Entries are added as junctions are crossed. By default, referrals live for 5 minutes. Client side PKT is maintained in RAM.

Host (server side):

One per Machine. Maintained in Registry.

Directory Service Store:

A PKT Object centrally stores all Dfs volume knowledge. Accessed and administrated by all machines participating in that Fault Tolerant volume.

Figure 5: Partition knowledge table

As shown in Figure 5, the partition knowledge table maps the Logical Dfs name space into physical referrals (alternate volumes appear as a list for a single junction).

When the Dfs client attempts to navigate a junction, it first looks to its locally cached PKT entries. If the referral cannot be resolved, the client contacts the Dfs root. If the referral still cannot be resolved, an error occurs. If the referral is properly resolved, the client adds the referral to its local table of entries.

When a Dfs client obtains a referral from the PKT, that referral is cached for five minutes. If the client reuses that referral, the time-to-live is renewed; otherwise, the cache expires. If alternate volumes exist for a particular referral, all alternates are handed to and cached by the client. The client randomly selects which referral to use.

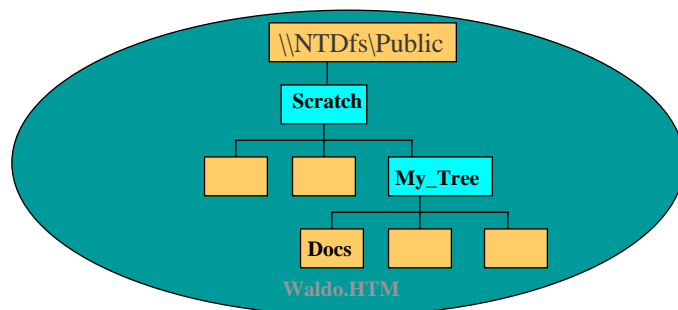
Dfs Referrals: How Junctions Are Resolved from Logical Names into Physical Addresses

The Dfs-aware redirector, SMB server, and Dfs driver collaborate to reroute path-based operations to the actual server/share hosting the file or directory. There are two new transaction SMBs defined for this, seven public administrative APIs, and 10

private APIs. These are covered in Appendix 2.

With a stand-alone Dfs volume, the referral process never extends more than one level deep before encountering either a leaf (a point which cannot junction, such as a down-level volume), or an inter-Dfs link. Upon encountering an inter-Dfs link, the referral process begins over on the other side of the junction to resolve referrals deeper in the logical name space.

The directory service-enabled version of Dfs will permit junctions to junctions (without requiring inter-Dfs links). In this situation, the referral process expressly searches for the longest (most “\” characters) referral that can be resolved from the requested path. This assures that the final destination is resolved with a single referral.



Get me: \\NTDfs\Public\Scratch\My_Tree\Docs\Waldo.HTM

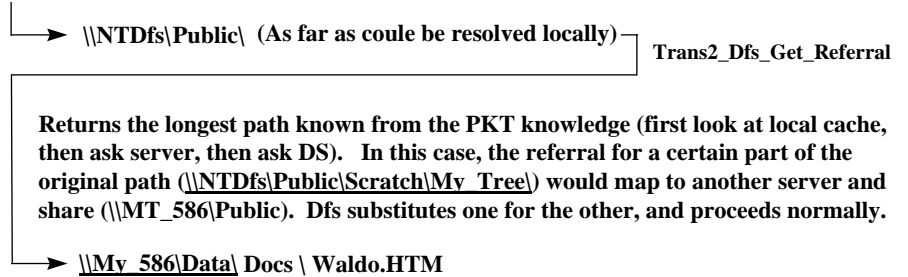


Figure 6: Trans2_Dfs_Get_Referral process

In Figure 6, the directories \Scratch and \My_Tree are meant to represent junctions to other servers. (For Windows NT 4.0, this is not possible without inter-Dfs links. In this case, two referral operations would be required, rather than just one.)

Fail-over Between Alternates Volumes

As outlined in the two sections above, referrals are cached locally for performance considerations, and, when alternates are available, all alternates are provided to the client. The client arbitrarily chooses which referral to use (in the directory-enabled version of Dfs, this is based on sites to ensure that clients select available alternates within their site).

Once a referral is selected among the alternates, a session setup is performed (credentials are passed to the new server if a prior connection does not exist). Should the selected referral fail, a fail-over process begins. The speed and implications of the fail-over are dependent on what the client was doing at the time of the failure and how the failure occurred.

Scenario #1

A client is browsing an alternate volume. The computer hosting the alternate loses power or drops completely from the network for any reason. To fail-over, the client must first detect that the hosting computer is no longer present. How long this takes depends on which protocol the client is using. Many protocols account for slow and loosely connected WAN links, and therefore may have retry counts of up to two minutes before the protocol itself times out. Once that occurs, Dfs immediately selects a new alternate. If none are available from the local cache, the Dfs client consults with the Dfs root to see if the administrator has modified any PKT entries. If no alternates are available at the root, a failure occurs; otherwise, Dfs initiates a fresh alternate selection and session setup.

Scenario #2

A client is browsing an alternate volume. The computer hosting the alternate loses a hard disk hosting the alternate, or the share is deactivated. In this situation, the server hosting the alternate is still responding to the client request; the fail-over to a fresh alternate is nearly instantaneous.

Scenario #3

A client has open files. The computer hosting the alternate loses power or drops completely from the network for any reason. In this scenario, the same protocol fail-over process described in Scenario #1 occurs, but the application that previously had file locks from the previous alternate must detect the change and establish new locks.

New attempts to open files trigger the same fail-over process described in Scenario #1. Operations on already open files fail with appropriate errors.

Scenario #4

A client has open files. The computer hosting the alternate loses a hard disk hosting the alternate, or the share is deactivated. In this scenario, the same very quick fail-over process described in Scenario #2 occurs, but the application that previously had file handles from the previous alternate must detect the change and establish new handles.

Security

Dfs allows special handling of security issues at session setup and with ACLs that are not consistent system-wide.

Session Setups

As each junction is crossed and cached for the first time, the Dfs-aware client establishes a session setup with the server on the other side of the junction. The

credentials that the user originally supplied to connect with Dfs are used (for example, Net Use * \\Server\Dfs_Share /u:domain\User). If the user did not supply credentials, the credentials cached when the user logged into his or her workstation are used.

ACLs

File ACLs are administered at each individual physical share. There is no mechanism to administrate ACLs system-wide from the Dfs root, nor is there an attempt to keep ACLs consistent between alternate volumes. There are several reasons for this:

- A centrally administrated logical ACL database could be bypassed, as users could Net Use directly to the physical resource.
- The logical Dfs volume can cross between FAT and NTFS volumes, as well as contain leaves from other network operating systems. There is no reasonable way to set an inherited Deny ACL which starts on an NTFS volume, passes to FAT, passes back to NTFS, and concludes on a NetWare volume.
- A tool that walks the logical name space, setting ACLs appropriately, would require a complicated message and transactioning engine to ensure that ACLs were queued and updated over loosely connected and/or unreliable networks.
- Storage quotas available in Windows 2000 require an additional burden of tallying storage for all possible users across all possible volumes to establish when users have exceeded their storage allotment.

Dfs binary component	Description
DfsAdmin.exe	Administrative UI tool
DfsSrv.exe	Dfs Server-side service (manages the Partition Knowledge Table)
DfsUI.dll	Calls used by DfsAdmin
Dfs.sys	Dfs driver used by the Dfs Service (for example, to hand out referrals)
DfsInit.exe	Boot time to initialize Dfs (Server only)
DfsSetup.dll	Network Control Panel (also used at install time)

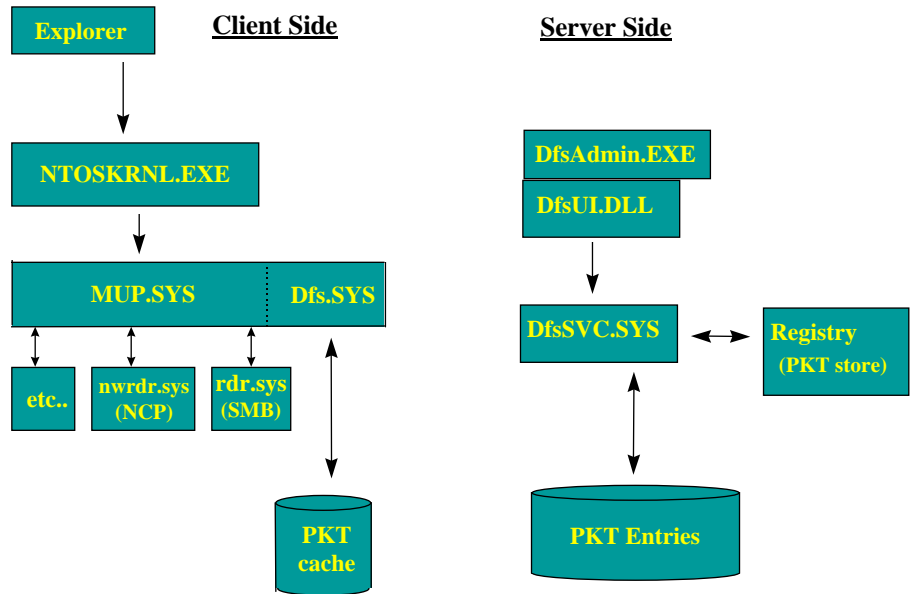


Figure 7: Block architecture showing various binary components

A utility called DfsCMD.exe also permits a Dfs to be modified with scripting commands.

Considerations for Developers Writing Applications that Use Dfs

This section provides suggestions concerning free space, connections, file systems, and backup and restoration for developers who are writing applications using Dfs.

Free Space

When enumerating the available storage space on a volume, **GetDiskFreeSpace** returns the space available at the root of the Dfs volume. It does not walk the tree, summing up the space available across assorted junctions. **GetDiskFreeSpaceEX** is a new function to return the space available at a specified point within a name space, which allows you to obtain the free space on the other side of a junction. This result depends on whether alternate volumes are mounted and alternates have identical storage devices.

Connections

Since Dfs allows workstations and servers to participate equally as junction points, a Windows NT workstation only allows only 10 clients to connect to it. This means that a share on a Windows NT workstation published in Dfs cannot be accessed once 10 connections have been established to that workstation (whether or not those connections use the private Dfs). Stand-alone servers do not have this limitation. Only Windows NT Server can host a Dfs root, but any share or volume accessible through the network can participate in the Dfs name space.

File Systems

Dfs is always hosted on Windows NT Server 4.0 or Windows 2000 Server, either in FAT or NTFS format. As junctions are crossed, you can change from one file format to another. With down-level volumes, you can also change from Windows NT to an alternate NOS that is formatted uniquely and has a different security model. Another effect of junction to different physical servers is that the network address can change between logical subdirectories (this is transparent to users and applications).

Backup and Restoration

With Dfs, a volume can be built to encompass all storage on your network. This allows all storage to be backed up through a single name space. Your backup solution must be aware of your Dfs topology. For example, if you wish to restore a portion of your logical name space when those physical servers no longer exist, you must be able to identify that fact, and you must be able to determine what to do with the data. In this situation, you could choose to queue the restore operation, on the assumption that the server is temporarily down, move the contents to another location, and automatically reconfigure the logical name space or cancel the operation.

TAKING ADVANTAGE OF DIRECTORY SERVICES

Dfs takes advantage of the following new features:

- Directory services for storing existing administrative information.
- LDAP as the protocol for updating and querying for Dfs information.
- Directory services store for providing root-level fault tolerance in Dfs.
- Directory services for keeping all participating computers in any Dfs root synchronized in their perception of the Dfs structure.
- Directory services site topology for providing intelligent replica selection.
- Content Replication Service for keeping alternate volumes synchronized with one another.

Fault Tolerant Dfs - Storing Dfs Knowledge in the DS

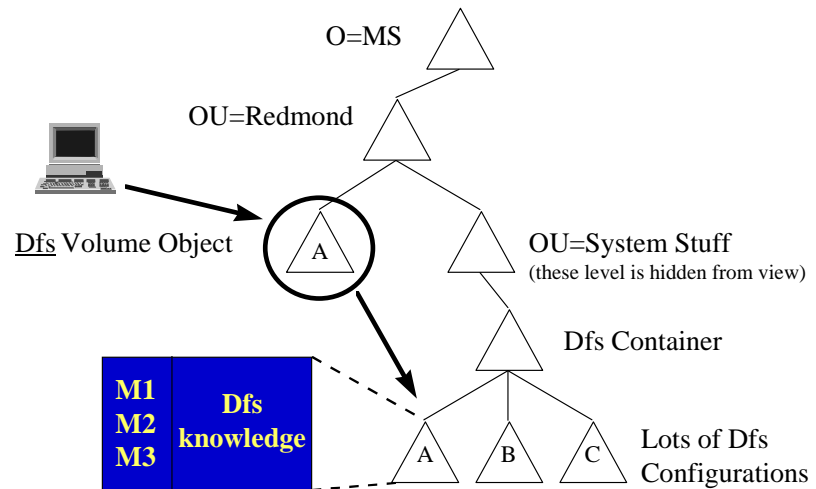


Figure 8: Information storage

For more information on directory service objects, see Appendix 3.

Public Interfaces

The Microsoft Platform SDK contains full documentation on these interfaces. This interface applies to both the Windows NT 4.0 and Windows 2000 versions of Dfs.

NetDfsAdd

This function either creates a new junction point, that is, a link to a server share, or adds shares to an existing junction point in a distributed file system (Dfs) tree structure.

NetDfsAdd(

```
LPWSTR DfsEntryPath,      // Dfs entry path for this added volume or storage
LPWSTR ServerName,       // Name of server exporting the storage
LPWSTR 0,                 // Existing share name for the storage
LPWSTR Comment,          // Optional comment for this volume or storage
DWORD Flags,             // Zero for no flags
);
```

Parameters

DfsEntryPath

[in] Pointer to a null-terminated Unicode character string that specifies the Universal Naming Convention (UNC) path name of a junction point in a Dfs tree structure. The string must be of the form:

```
Dfsname\sharename\path-to-junction-point
```

where *Dfsname* is the name of a Windows NT server that hosts the Dfs root volume, *sharename* is the name of a share that is published on the Dfs host server, and *path-to-junction-point* specifies the UNC network path name to a physical shared volume.

ServerName

[in] Pointer to a null-terminated Unicode character string that specifies the name of the storage server that the junction point references.

ShareName

[in] Pointer to a null-terminated Unicode character string that specifies the name of the share on the storage server that the junction point references.

Comment

[in] Pointer to a null-terminated Unicode character string that contains an optional comment associated with the junction point.

Flags

[in] Specifies a flag value. If you set this parameter to the value `DFS_ADD_VOLUME` this function fails if the junction point already exists. If you set this parameter to 0, this function either creates a new junction point if one does not exist, or it adds a new share to an existing junction point.

Return Values

NERR_Success indicates success. A Win32 API error value indicates failure. See Error Codes in the Appendix for a list of error values.

See Also: **NetDfsEnum**, **NetDfsRemove**

NetDfsRemove

This function removes a server share from a junction point in a distributed file system (Dfs) tree structure. If the specified share is the last share associated with the junction point, this function also removes the junction point.

NetDfsRemove(

```
LPWSTR DfsEntryPath,      // Dfs entry path for this volume or storage
LPWSTR ServerName,        // Name of server exporting the storage
LPWSTR ShareName,         // Name of share exporting the storage
);
```

Parameters

DfsEntryPath

[in] Pointer to a null-terminated Unicode character string that specifies the Universal Naming Convention (UNC) path name of a junction point in a Dfs tree structure. The string must be of the form:

```
Dfsname\sharename\path-to-junction-point
```

where *Dfsname* is the name of a Windows NT server that hosts the Dfs root volume; *sharename* is the name of a share that is published on the Dfs host server; and *path-to-junction-point* specifies the UNC network path name to a physical shared volume.

ServerName

[in] Pointer to a null-terminated Unicode character string that specifies the name of the storage server that the junction point references.

ShareName

[in] Pointer to a null-terminated Unicode character string that specifies the name of the share on the storage server that the junction point references.

Return Values

NERR_Success indicates success. A Win32 API error value indicates failure. See Error Codes in the Appendix for a list of error values.

See Also: **NetDfsAdd**, **NetDfsEnum**

NetDfsEnum

This function enumerates all the junction points in the named distributed file system (Dfs) tree structure. It returns information about the junction points, based on the

level of information specified by the *Level* parameter.

NetDfsEnum(

```
LPWSTR DfsName,           // Name of the Dfs for enumeration
DWORD Level,              // Level of information requested
DWORD PrefMaxLen,         // Advisory, but -1 means "get it all"
LPBYTE* Buffer,            // API allocates and returns buffer with requested
                           // info
LPDWORD EntriesRead,      // Number of entries returned
LPDWORD ResumeHandle,     // Must be 0 on first call, reused on subsequent calls
);
```

Parameters

DfsName

[in] Pointer to a null-terminated Unicode character string that specifies the name of a Windows NT server that hosts the Dfs root volume.

Level

[in] Specifies the information level of the request. This parameter can be one of the following values:

Value	Description
1	Return Dfs volume names. The <i>buffer</i> parameter contains an array of DFS_INFO_1 structures.
2	Return Dfs volume names and volume information. The <i>buffer</i> parameter contains an array of DFS_INFO_2 structures.
3	Return Dfs names, volume information, and network path information. The <i>buffer</i> parameter contains an array of DFS_INFO_3 structures.

PrefMaxLen

[in] Specifies the preferred maximum number of bytes that should be returned by this enumeration function call.

Buffer

[out] Pointer to the address of a buffer that contains the requested information structures.

EntriesRead

[out] Pointer to a **DWORD** that contains the actual enumerated junction point count.

ResumeHandle

[in/out] Pointer to a **DWORD** that contains a handle that is used to continue the enumeration. The handle should be 0 on the first call and left unchanged for subsequent calls.

Return Values

NERR_Success indicates success. A Win32 API error value indicates failure. See Error Codes in the Appendix for a list of error values.

Remarks

Call this function with the *ResumeHandle* parameter set to 0 to begin the enumeration. To retrieve information about additional junction points, call the function with the *ResumeHandle* returned by the previous call to this function.

This function allocates the memory required for the information structure buffer. The size of the memory can be greater than the amount specified by the *PrefMaxLen* parameter.

See Also: **DFS_INFO_1**, **DFS_INFO_2**, **DFS_INFO_3**, **DFS_STORAGE_INFO**, **NetDfsAdd**, **NetDfsRemove**

NetDfsGetInfo

This function retrieves information about a junction point in the named distributed file system (Dfs) tree structure. It can return information specific to a server and share or information specific to an entire junction point.

NetDfsGetInfo(

```
    LPWSTR DfsEntryPath,           // Dfs entry path for the volume
    LPWSTR ServerName              // Name of server exporting the storage
OPTIONAL,
    LPWSTR ShareName              // Name of share exporting the storage
OPTIONAL,
    DWORD Level,                  // Level of information requested
    LPBYTE* Buffer,                // API allocates and returns buffer with
                                  // requested information
);
```

Parameters

DfsEntryPath

[in] Pointer to a null-terminated Unicode character string that specifies the Universal Naming Convention (UNC) path name of a junction point in a Dfs tree structure. The string must be of the form:

```
Dfsname\sharename\path-to-junction-point
```

where *Dfsname* is the name of a Windows NT server that hosts the Dfs root volume. *sharename* is the name of a share that is published on the Dfs host server, and *path-to-junction-point* specifies the UNC network path name to a physical shared volume.

ServerName

[in] Pointer to a null-terminated Unicode character string that specifies the name of the storage server that the junction point references. This parameter is

optional.

ShareName

[in] Pointer to a null-terminated Unicode character string that specifies the name of the share on the storage server that the junction point references. This parameter is optional.

Level

[in] Specifies the information level of the request. This parameter can be one of the following values:

Value	Description
1	Return Dfs volume names. The <i>buffer</i> parameter contains an array of DFS_INFO_1 structures.
2	Return Dfs volume names and volume information. The <i>buffer</i> parameter contains an array of DFS_INFO_2 structures.
3	Return Dfs names, volume information, and network path information. The <i>buffer</i> parameter contains an array of DFS_INFO_3 structures.
100	Return a comment about this Dfs volume or server. The <i>buffer</i> parameter contains a DFS_INFO_100 structure.

Buffer

[out] Pointer to the address of a buffer that contains the requested information structures.

Return Values

NERR_Success indicates success. A Win32 API error value indicates failure. See Error Codes in the Appendix for a list of error values.

Remarks

If you specify both the *ServerName* and *ShareName* parameters, this function returns information specific to that server and share. If the parameters are not specified, this function returns information that is specific to the entire junction point.

See Also: **DFS_INFO_1**, **DFS_INFO_2**, **DFS_INFO_3**, **DFS_INFO_100**, **DFS_STORAGE_INFO**, **NetDfsEnum**

NetDfsSetInfo

This function associates information with a junction point in the named distributed file system (Dfs) tree structure. This function can set information relevant to a specific server and share or information specific to an entire junction point.

NetDfsSetInfo(

```
LPWSTR DfsEntryPath,           // Dfs entry path for the volume
LPWSTR ServerName OPTIONAL,    // Name of server exporting the storage
LPWSTR ShareName OPTIONAL,     // Name of share exporting the storage
DWORD Level,                   // Level of information to be set
```

```
LPBYTE Buffer, // Buffer holding information
);
```

Parameters

DfsEntryPath

[in] Pointer to a null-terminated Unicode character string that specifies the Universal Naming Convention (UNC) path name of a junction point in a Dfs tree structure. The string must be of the form:

```
Dfsname\sharename\path-to-junction-point
```

where *Dfsname* is the name of a Windows NT server that hosts the Dfs root volume, *sharename* is the name of a share that is published on the Dfs host server, and *path-to-junction-point* specifies the UNC network path name to a physical shared volume.

ServerName

[in] Pointer to a null-terminated Unicode character string that specifies the name of the storage server that the junction point references. This parameter is optional.

ShareName

[in] Pointer to a null-terminated Unicode character string that specifies the name of the share on the storage server that the junction point references. This parameter is optional.

Level

[in] Specifies the information level of the set request. This parameter can be only the following value:

Value	Description
100	Return a comment about this Dfs volume or server. The <i>buffer</i> parameter contains a DFS_INFO_100 structure.

buffer

[in] Pointer to a buffer that contains the information structure.

Return Values

NERR_Success indicates success. A Win32 API error value indicates failure. See Error Codes in the Appendix for a list of error value.

Remarks

If you specify both the *ServerName* and *ShareName* parameters, this function returns information specific to that server and share. If the parameters are not specified, this function returns information that is specific to the entire junction point.

See Also: **DFS_INFO_100**, **NetDfsEnum**

Dfs API Error Values

Value	Description
NERR_DfsInternalCorruption (NERR_BASE+560)	The internal database that the Dfs Service maintains is corrupt.
NERR_DfsVolumeDataCorrupt (NERR_BASE+561)	One of the records in the internal database that the Dfs Service maintains is corrupt.
NERR_DfsNoSuchVolume (NERR_BASE+562)	There is no volume that matches the DfsEntryPath parameter.
NERR_DfsVolumeAlreadyExists (NERR_BASE+563)	A Dfs volume with the specified name already exists.
NERR_DfsAlreadyShared (NERR_BASE+564)	The server share specified is already shared.
NERR_DfsNoSuchShare (NERR_BASE+565)	The indicated server share does not support the indicated volume.
NERR_DfsNotALeafVolume (NERR_BASE+566)	The operation is not valid on a non-leaf Dfs volume.
NERR_DfsLeafVolume (NERR_BASE+567)	The operation is not valid on a Dfs leaf volume.
NERR_DfsVolumeHasMultipleServers(NERR_BASE+568)	The Dfs Service is unable to complete this operation because the volume has multiple servers.
NERR_DfsCantCreateJunctionPoint (NERR_BASE+569)	The Dfs Service is unable to create this junction point.
NERR_DfsServerNotDfsAware (NERR_BASE+570)	The server is not Dfs-aware either because Dfs server software has not been installed or because the Distributed File Service has been terminated manually.
NERR_DfsBadRenamePath (NERR_BASE+571)	The specified rename target path is invalid.
NERR_DfsVolumeIsOffline (NERR_BASE+572)	The specified Dfs volume is offline.
NERR_DfsNoSuchServer (NERR_BASE+573)	The specified server is not a server for this Dfs volume.
NERR_DfsCyclicalName (NERR_BASE+574)	A cycle in the Dfs name was detected.
NERR_DfsNotSupportedInServerDfs (NERR_BASE+575)	This operation is not supported on a server-based Dfs; it is only supported on a domain-based Dfs.
NERR_DfsInternalError (NERR_BASE+590)	A Dfs internal error has occurred.

APPENDIX 2: CIFS SPECIFICATION

The Common Internet File System (CIFS) defines a standard remote file system access protocol for use over the Internet, enabling groups of users to work together and share documents across the Internet or within their corporate intranets. CIFS is an open, cross-platform technology based on the native file-sharing protocols built into Windows and other popular operating systems, and supported on numerous other platforms, including UNIX. Dfs is covered as part of the CIFS specification.

Dfs Support

Protocol dialects of Windows NT LAN Manager 0.12 and later support distributed file system operations. The Distributed File System allows this protocol to use a single consistent file-naming scheme that may span multiple servers and shares. The distributed file system model employed is a referral-based model. This protocol specifies the manner in which clients receive referrals.

The client can set a flag in the request SMB header, indicating that the client wants the server to resolve the SMB paths within the Dfs known to the server. The server attempts to resolve the requested name to a file contained within the local directory tree indicated by the TID of the request and proceeds normally. If the request path name resolves to a file on a different system, the server returns `STATUS_DFS_PATH_NOT_COVERED`.

The server does not support the part of the Dfs name space needed to resolve the path name in the request. The client should request a referral from this server for further information.

A client asks for a referral with the `TRANS2_DFS_GET_REFERRAL` request containing the Dfs pathname of interest. The response from the server indicates how the client should proceed. The method by which the topological knowledge of the Dfs is stored and maintained by the servers is not specified by this protocol.

Dfs Path Names

A Dfs pathname adheres to the standard described in File Names. A Dfs-enabled client accessing a Dfs share should set the `FLAGS2` bit 12 in all name-based SMB headers, indicating to the server that the enclosed path name should be resolved in the Distributed File System name space. The path name should always have the full file name, including the server name and share name. If the server can resolve the Dfs name to a piece of local storage, the local storage is accessed. If the server determines that the Dfs name actually maps to a different server share, the access to the name fails with the following error:

```
STATUS_PATH_NOT_COVERED (SMB Status code 0xC0000257)
```

On receiving this error, the Dfs enabled client should ask the server for a *REFERRAL* (see `TRANS2_GET_DFS_REFERRAL`). The referral request should contain the full file name.

The response to the request contains a list of server and share names to try, and

the part of the request file name that is joined to the list of server shares. If the `ServerType` field of the referral is set to 1 (SMB server), the client should resubmit the request with the *ORIGINAL* file name to one of the server shares in the list, once again setting the *FLAGS2* bit 12 bit in the SMB. If the `ServerType` field is not 1, the client should strip off the part of the file name that joins to the server share before resubmitting the request to one of the servers in the list.

A response to a referral request may elicit a response that does not have the `StorageServers` bit set. In that case, the client should resubmit the *REFERRAL REQUEST* to one of the servers in the list, until it obtains a referral response that has the `StorageServers` bit set, at which point the client can resubmit the request SMB to one of the listed server shares.

If, after getting a referral with the `StorageServers` bit set and resubmitting the request to one of the server shares in the list, the server fails the request with `STATUS_PATH_NOT_COVERED`, there is an inconsistency between the view of the Dfs name space held by the server granting the referral and the server listed in that referral. In this case, the client may inform the server granting the referral of this inconsistency with the *TRANS2_REPORT_DFS_INCONSISTENCY* SMB.

TRANS2_GET_DFS_REFERRAL:

Retrieve Distributed File-system Referral

The client sends this request to ask the server to convert *REQUESTFILENAME* into an alternate name for this file. This request can be sent to the server if the server response to the `NEGOTIATE` SMB included the `CAP_DFS` capability. The TID of the request must be `IPC$`. *BIT15* of *FLAGS2* in the SMB header must be set, indicating that this is a `UNICODE` request.

Client Request	Description
WordCount	15
TotalDataCount	0
SetupCount	1
Setup[0]	TRANS2_GET_DFS_REFERRAL
Parameter Block Encoding	Description
USHORT MaxReferralLevel	Latest referral version number understood
WCHAR RequestFileName;	DFS name of file for which referral is sought

Response Data Block	Description
USHORT PathConsumed;	Number of REQUESTFILENAME bytes client
USHORT NumberOfReferrals;	Number of referrals contained in this response
USHORT Flags;	bit 0: The servers in REFERRALS are capable of fielding TRANS2_GET_DFS_REFERRAL. bit 1: The servers in REFERRALS should hold the storage for the requested file.
REFERRAL_LIST Referrals[]	Set of referrals for this file
UNICODESTRINGE Strings	Used to hold the strings pointed to by version 2 Referrals in REFERRALS.

The server response is a list of *REFERRALS* that tell the client where to resubmit the request to obtain access to the file. *PATHCONSUMED* in the response indicates to the client how many characters of *REQUESTFILENAME* have been consumed by the server. When the client chooses one of the referrals to use for file access, the client may need to strip the leading *PATHCONSUMED* characters from the front of *REQUESTFILENAME* before submitting the name to the target server. Whether or not the pathname should be trimmed is indicated by the individual referral, as detailed below.

FLAGS indicates how this referral should be treated. If *BIT0* is clear, any entity in the *REFERRALS* list holds the storage for *REQUESTFILENAME*. If *BIT0* is set, any entity in the *REFERRALS* list has further referral information for *REQUESTFILENAME* – a *TRANS2_GET_DFS_REFERRAL* request should be sent to an entity in the *REFERRALS* list for further resolution.

The format of an individual referral contains version and length information allowing the client to skip referrals that it does not understand. *MaxReferralLevel* indicates to the server the latest version of a referral that the client can digest. Since each referral has a uniform element, *MAXREFERRALLEVEL* is advisory only. Each element in *REFERRALS* has this envelope:

REFERRAL_LIST Element	
USHORT VersionNumber	Version of this referral element
USHORT ReferralSize	Size of this referral element

The following referral element versions are defined:

Version 1 Referral Element Format

USHORT ServerType	Type of NODE handling referral: <ul style="list-style-type: none">• Don't know• SMB Server• Netware Server• Domain
USHORT ReferralFlags	Flags which describe this referral: 01 - Strip off PATHCONSUMED characters before submitting REQUESTFILENAME to NODE
UNICODESTRING Node	Name of entity to visit next

Version 2 Referral Element Format

USHORT ServerType	Type of NODE handling referral: <ul style="list-style-type: none">• Don't know• SMB Server• Netware Server• Domain
USHORT ReferralFlags	Flags which describe this referral: 01 - Strip off PATHCONSUMED characters before submitting REQUESTFILENAME to NODE
ULONG Proximity	A hint describing the proximity of this server to the client. 0 indicates the closest, higher numbers indicate increasingly "distant" servers. The number is only relevant within the context of the servers listed in THIS particular SMB.
ULONG TimeToLive	Number of seconds for which the client can cache this referral.
USHORT DfsPathOffset	Offset, in bytes from the beginning of this referral, of the Dfs Path that matched PATHCONSUMED bytes of the REQUESTFILENAME.
USHORT DfsAlternatePathOffset	Offset, in bytes from the beginning of this referral, of an alternate name (8.3 format) of the Dfs Path that matched PATHCONSUMED bytes of the REQUESTFILENAME.
USHORT NetworkAddressOffset	Offset, in bytes from the beginning of this referral, of the entity to visit next.

The SMB protocol imposes no referral selection policy.

TRANS2_REPORT_DFS_INCONSISTENCY: Inform a Server About Dfs Error

As part of the Distributed Name Resolution algorithm, a Dfs client may discover a knowledge inconsistency between the referral server (the server that handed out a referral), and the storage server (the server to which the client was redirected to by the referral server). When such an inconsistency is discovered, the Dfs client optionally sends this SMB to the referral server, allowing the referral server to take corrective action.

Client Request	Description
WordCount	15
MaxParameterCount	0
SetupCount	1
Setup[0]	TRANS2_REPORT_DFS_INCONSISTENCY
Parameter Block Encoding	Description
UNICODESTRING RequestFileName;	DFS Name of file for which referral was sought

The data part of this request contains the referral element (version 1 format only) believed to be in error. These are encoded as described in the `TRANS2_GET_DFS_REFERRAL` response. If the server returns success, the client can resubmit the `TRANS2_GET_DFS_REFERRAL` request to this server to get a new referral. It is not mandatory for the Dfs knowledge to be automatically repaired; the client must be prepared to receive further errant referrals and must not wind up looping between this request and the `TRANS2_GET_DFS_REFERRAL` request.

BIT15 of *FLAGS2* in the SMB header must be set, indicating this is a UNICODE request.

APPENDIX 3: DIRECTORY SERVICE OBJECTS

The following outlines the structure maintained in the directory to store Dfs related knowledge. These objects can all be manipulated with the Public API described in Appendix 1

Dfs-Configuration

This object is a holder of all Dfs containers within an organization.

```
Common-Name=Dfs-Configuration
Admin-Display-Name=Dfs-Configuration
Admin-Description=Dfs-Configuration
Object-Class=Class-Schema
Comment:Holds all Fault-tolerant DFS configurations
LDAP-Display-Name=dfsConfiguration
Governs-ID=1.2.840.113556.1.5.42
Structural Class
Rdn-Att-Id=Common-Name
Schema-ID-GUID={8447f9f2-1027-11d0-a05f-00aa006c33ed}
Default-Security-Descriptor=O:CO G:CG
D:(A;GA;;;DA)(A;GA;;;SY)(A;GA;;;CO)
Subclass of:Container
System-Only=FALSE
Poss-Superiors=Configuration
```

FT-Dfs

This is an individual Dfs container object. It holds the metadata of all root-level participating computers as well as all junction information.

```
Common-Name=FT-Dfs
Admin-Display-Name=FT-Dfs
Admin-Description=FT-Dfs
Object-Class=Class-Schema
Comment:Defines a single Fault-tolerant DFS configuration - MilanS
LDAP-Display-Name=fTDfs
Governs-ID=1.2.840.113556.1.5.43
Structural Class
Rdn-Att-Id=Common-Name
Schema-ID-GUID={8447f9f3-1027-11d0-a05f-00aa006c33ed}
Default-Security-Descriptor=O:CO G:CG
D:(A;GA;;;DA)(A;GA;;;SY)(A;GA;;;CO)
Subclass of:Container
System-Only=FALSE
Poss-Superiors=Dfs-Configuration
Must Contain:PKT
Must Contain:PKT-Guid
Must Contain:Remote-Server-Name
```

PKT

This object contains a table of all the junction knowledge for a particular Dfs volume. The **NetDfsEnum** function retrieves this information.

```
Common-Name=PKT
Admin-Display-Name=PKT
Admin-Description=PKT
Object-Class=Attribute-Schema
Attribute-ID=1.2.840.113556.1.4.206
OM-Syntax=4
Attribute-Syntax=String(Octet)
LDAP-Display-Name=pKT
Single-valued
System-Only=FALSE
Schema-ID-GUID={8447f9f1-1027-11d0-a05f-00aa006c33ed}
```

PKT-Guid

System Level object use administratively to ensure consistency between computers hosting roots to any given fault tolerant Dfs volume.

```
Common-Name=PKT-Guid
Admin-Display-Name=PKT-Guid
Admin-Description=PKT-Guid
Object-Class=Attribute-Schema Attribute-ID=1.2.840.113556.1.4.205
OM-Syntax=4
Attribute-Syntax=String(Octet)
LDAP-Display-Name=pKTGuid
Single-valued
System-Only=FALSE
Range 16 to 16
Schema-ID-GUID={8447f9f0-1027-11d0-a05f-00aa006c33ed}
```

Remote-Server-Name

Multivalue list of all computers participating at the root of a specific fault-tolerant Dfs.

```
Common-Name=Remote-Server-Name
Admin-Display-Name=Remote-Server-Name
Admin-Description=Remote-Server-Name
Object-Class=Attribute-Schema Attribute-ID=1.2.840.113556.1.4.105
OM-Syntax=64
Attribute-Syntax=String(Unicode)
LDAP-Display-Name=remoteServerName
Multi-valued
System-Only=FALSE
Schema-ID-GUID={bf967a12-0de6-11d0-a285-00aa003049e2}
```

FOR MORE
INFORMATION

For the latest information on Windows NT and Windows 2000, visit the Web site at <http://www.microsoft.com/ntserver/>, the Windows NT Server Forum on MSN™, and The Microsoft Network online service (GO WORD: MSNTS).