

Novell Developer Kit

www.novell.com

February 28, 2007

MULTIPLE AND INTER-FILE
SERVICES

N

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export, or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to www.novell.com/info/exports/ for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 1993-2007 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.novell.com/company/legal/patents/> and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell developer products, and to get updates, see developer.novell.com/ndk. To access online documentation for Novell products, see www.novell.com/documentation.

Novell Trademarks

For a list of Novell trademarks, see [Trademarks \(http://www.novell.com/company/legal/trademarks/tmlist.html\)](http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Contents

About This Guide	15
1 Data Migration Concepts	17
1.1 Support Module Information	17
1.2 Volume Information	18
1.3 Data Migration Functions	18
2 Data Migration Functions	19
NWGetDataMigratorInfo	20
NWGetDefaultSupportModule	22
NWGetDMFileInfo	24
NWGetDMVolumeInfo	27
NWGetSupportModuleInfo	30
NWMoveFileFromDM	32
NWMoveFileToDM	34
NWSetDefaultSupportModule	37
3 Data Migration Structures	39
SUPPORT_MODULE_IDS	40
SUPPORT_MODULE_INFO	41
4 Deleted File Concepts	43
4.1 Deleted File on NetWare 3.11 and above Servers	43
4.2 Deleted File Functions	43
5 Deleted File Functions	45
NWPurgeDeletedFile	46
NWRrecoverDeletedFile	49
NWRrecoverDeletedFileExt	52
NWScanForDeletedFiles	54
NWScanForDeletedFilesExt	57
6 Deleted File Structures	59
NWDELETED_INFO	60
NWDELETED_INFO_EXT	64
7 File Engine Functions	67
CountComponents	68
FEConvertDirectoryNumber	70
FEcreat	72
FEFlushWrite	74

FEGetCWDnum	75
FEGetCWVnum	76
FEGetEntryVersion	77
FEGetOpenFileInfo	79
FEGetOpenFileInfoForNS	82
FEGetOriginatingNameSpace	85
FEMapConnsHandleToVolAndDir	87
FEMapHandleToVolumeAndDirectory	89
FEMapPathVolumeDirToVolumeDir	90
FEMapVolumeAndDirectoryToPath	92
FEMapVolumeAndDirectoryToPathForNS	94
FEMapVolumeNumberToName	96
FEQuickClose	97
FEQuickFileLength	99
FEQuickOpen	101
FEQuickRead	103
FEQuickWrite	105
FERegisterNSPathParser	107
FESetCWDnum	109
FESetCWVandCWDnums	110
FESetCWVnum	111
FESetOriginatingNameSpace	112
FEsopen	114

8 File System Concepts 117

8.1 Directory Entries	117
8.1.1 Directory Entry Information	117
8.1.2 Directory Entry Information Access	118
8.1.3 Directory Entry Attributes	118
8.1.4 Directory Entry Functions	119
8.1.5 Directory Information Functions	119
8.2 Directory Handles	120
8.2.1 Directory Handle Functions	120
8.3 File and Directory Paths	120
8.3.1 Wildcard Characters	121
8.3.2 Search Attributes	121
8.3.3 UTF-8 Path and Filenames	121
8.4 File Access	123
8.5 File I/O	123
8.6 Inheritance	123
8.7 Effective Rights	124
8.8 Trustees	124
8.8.1 Trustee Rights	124
8.8.2 Trustee Functions	125
8.9 NLM File Information	126
8.9.1 File Attributes	127
8.9.2 Extended File Attributes	128
8.9.3 Directory Entry Table	128
8.9.4 Volume Table	129
8.10 Directory Task Functions	129
8.11 Directory Space Functions	129
8.12 File Handle Conversion Functions	129
8.13 File Information Functions	129

8.14	File Task Functions	130
8.15	File Usage Functions	130
9	File System Tasks	131
9.1	Directory-Based Tasks	131
9.1.1	Allocating a Directory Handle	131
9.1.2	Accessing a Directory Handle	131
9.1.3	Combining a Path and Directory Handle	131
9.1.4	Accessing File Information for 3.11 and Above	132
9.2	File-Based Tasks	132
9.2.1	Locating Files	132
9.2.2	Converting File Handles	132
9.2.3	Deleting Files	133
9.3	Disk Space Management Tasks	133
9.3.1	Limiting Directory Space	133
9.3.2	Monitoring File Usage	133
9.4	Trustee Tasks	133
9.4.1	Adding and Deleting File System Trustees	134
9.4.2	Scanning File System Trustees	134
9.5	NLM-Based Tasks	134
9.5.1	Accessing Files on a Server (NLM)	134
9.5.2	Purging and Salvaging Files (NLM)	135
10	File System Functions	137
10.1	A*-M* Functions	137
	access	138
	chdir	140
	chmod	141
	closedir	143
	FileServerFileCopy	144
	getcwd	146
	GetExtendedFileAttributes	147
	_makepath	149
	mkdir	151
10.2	NWA*-NWF* Functions	151
	NWAddTrustee	153
	NWAddTrusteeExt	156
	NWAddTrusteeToDirectory	158
	NWAllocPermanentDirectoryHandle	161
	NWAllocTemporaryDirectoryHandle	163
	NWConvertFileHandle	166
	NWConvertHandle	168
	NWCreateDirectory	170
	NWDeallocateDirectoryHandle	173
	NWDeleteDirectory	175
	NWDeleteTrustee	177
	NWDeleteTrusteeExt	179
	NWDeleteTrusteeFromDirectory	181
	NWFileServerFileCopy	183
10.3	NWGet* Functions	185
	NWGetCompressedFileLengths	186
	NWGetDirectoryEntryNumber	188
	NWGetDirectoryHandlePath	191
	NWGetDirSpaceInfo	193
	NWGetDirSpaceLimitList	195

	NWGetDirSpaceLimitList2	197
	NWGetDiskIOsPending	199
	NWGetEffectiveRights	200
	NWGetEffectiveRightsExt.	203
	NWGetExtendedFileAttributes2	206
	NWGetFileConnectionID	209
	NWGetFileDirEntryNumber	211
	NWGetSparseFileBitMap	214
	NWGetVolumeFlags	216
10.4	NWI*-NWR* Functions	217
	NWIntEraseFiles	218
	NWIntFileSearchContinue	221
	NWIntFileSearchInitialize	224
	NWIntMoveDirEntry	226
	NWIntScanDirectoryInformation2	229
	NWIntScanDirEntryInfo	232
	NWIntScanExtendedInfo	235
	NWIntScanFileInformation2	238
	NWIntScanFileInformation2Ext	241
	NWIntScanForTrustees	244
	NWIntScanForTrusteesExt.	248
	NWModifyMaximumRightsMask	251
	NWRenameDirectory	254
	NWRenameFile	256
10.5	NWS*-NWZ* Functions	258
	NWScanConnectionsUsingFile	260
	NWScanDirectoryForTrustees2	262
	NWScanOpenFilesByConn2	265
	NWSetCompressedFileLengths	267
	NWSetCompressedFileSize	269
	NWSetDirectoryHandlePath	271
	NWSetDirectoryInformation	274
	NWSetDirEntryInfo	277
	NWSetDirSpaceLimit	281
	NWSetExtendedFileAttributes2	283
	NWSetFileAttributes	286
	NWSetFileInformation2	289
	NWSetVolumeFlags	292
	NWVolumelsCDROM	294
10.6	O*-Z* Functions	295
	opendir	296
	PurgeErasedFile	298
	readdir	300
	remove	302
	rename	304
	rmdir	306
	SalvageErasedFile	307
	ScanErasedFiles	309
	SetExtendedFileAttributes	311
	SetFileInfo	313
	SetReaddirAttribute	316
	_splitpath	318
	stat	320
	tmpnam	322
	umask	323
	UnAugmentAsterisk	324
	unlink	325
	UseAccurateCaseForPaths	326

utime	327
11 File System Structures	329
CONN_USING_FILE	330
CONNS_USING_FILE	332
DIR	334
DIR_SPACE_INFO	337
ModifyStructure	339
NW_EXT_FILE_INFO	341
NW_FILE_INFO2	345
NW_FILE_INFO2_EXT	347
NW_LIMIT_LIST	349
NWDIR_INFO	351
NWENTRY_INFO	353
NWET_INFO	355
NWET_INFO_EXT	356
NWFILE_INFO	357
OPEN_FILE_CONN	359
OPEN_FILE_CONN_CTRL	362
SEARCH_DIR_INFO	363
SEARCH_FILE_INFO	366
stat	368
TRUSTEE_INFO	371
utimbuf	372
VOLUME_STATS	373
VOLUME_INFO	375
12 File System Monitoring Concepts	377
12.1 Registering for Callback	377
12.2 File Monitoring	377
12.2.1 Pre-Execution and Post-Execution Monitoring	378
12.2.2 Pre-Execution Callbacks	379
12.2.3 Post-Execution Callbacks	379
12.2.4 Callback Structures	379
12.3 Potential Uses	380
12.3.1 Hot Backup	381
12.3.2 Version Control	381
12.4 File System Monitoring Functions	381
13 File System Monitoring Tasks	383
13.1 Writing a File System Monitor NLM	383
14 File System Monitoring Functions	385
NWAddFSMonitorHook	386
NWRemoveFSMonitorHook	389
15 File System Monitoring Structures	391
CloseFileCallBackStruct	392

CreateDirCallBackStruct	393
CreateFileCallBackStruct	395
CreateAndOpenCallBackStruct	397
DeleteDirCallBackStruct	399
EraseFileCallBackStruct	400
GenericEraseFileCBStruct	402
GenericModifyDOSInfoCBStruct	404
GenericModifyNSInfoCBStruct	406
GenericOpenCreateCBStruct	408
GenericPurgeDeletedCBStruct	411
GenericRenameCBStruct	412
GenericSalvageDeletedCBStruct	414
ModifyDirEntryCallBackStruct	415
OpenFileCallBackStruct	418
PurgeDeletedCallBackStruct	421
RenameMoveEntryCallBackStruct	422
RenameNSEntryCallBackStruct	424
SalvageDeletedCallBackStruct	426
16 Name Space Concepts	427
16.1 Naming Conventions	427
16.2 Default Name Space	428
16.3 Primary Entry Information	428
16.3.1 Primary Entry Information Functions	430
16.4 Name Space Specific Information	430
16.4.1 Name Space Entry Bit Mask	431
16.4.2 Name Space Bit Mask	431
16.4.3 DOS Name Space Bit Mask	431
16.4.4 Name Space Specific Information Functions	432
16.5 Long to DOS Conversions	432
16.5.1 NetWare 4.x	432
16.5.2 NetWare 5.x and 6.x	434
16.6 General Name Space Functions	434
17 Name Space Tasks	437
17.1 Accessing Huge Name Space Information	437
18 Name Space Functions	439
18.1 Get* and Set* Functions	439
GetDataStreamName	440
GetNameSpaceName	442
SetCurrentNameSpace	444
SetTargetNameSpace	446
18.2 NWA* through NWI* Functions	446
NWAddTrusteeToNSDirectory	448
NWAllocTempNSDirHandle2	451
NWAllocTempNSDirHandle2Ext	453
NWDeleteNSEntry	455
NWDeleteNSEntryExt	457
NWDeleteTrusteeFromNSDirectory	459
NWGetDirectoryBase	461

NWGetDirectoryBaseExt	464
NWGetHugeNSInfo	466
NWGetLongName	468
NWGetLongNameExt	470
NWGetNameSpaceEntryName	472
NWGetNSEntryInfo	474
NWGetNSEntryInfoExt	477
NWGetNSFileDirEntryNumber	479
NWGetNSInfo	481
NWGetNSInfo (NLM)	483
NWGetNSLoadedList	485
NWGetNSLoadedList (NLM)	487
NWGetNSPath	489
NWGetNSPathExt	491
NWGetOwningNameSpace	493
NWIsLNSSupportedOnVolume	495
18.3 NWN* through NWW* Functions	496
NWNSGetDefaultNS	498
NWNSGetMiscInfo	500
NWNSRename	502
NWNSRenameExt	505
NWOpenCreateNSEntry	508
NWOpenCreateNSEntryExt	510
NWOpenDataStream	512
NWOpenNSEntry	516
NWOpenNSEntryExt	519
NWQueryNSInfoFormat	522
NWReadExtendedNSInfo	524
NWReadNSInfo	526
NWReadNSInfoExt	528
NWScanNSDirectoryForTrustees	530
NWScanNSEntryInfo	533
NWScanNSEntryInfoExt	536
NWScanNSEntryInfo2	538
NWScanNSEntryInfoSet	541
NWSetHugeNSInfo	544
NWSetLongName	546
NWSetNameSpaceEntryName	549
NWSetNSEntryDOSInfo	551
NWSetNSEntryDOSInfoExt	554
NWSetNSInfo	557
NWWriteExtendedNSInfo	559
NWWriteNSInfo	561
NWWriteNSInfoExt	563

19 Name Space Structures 565

MODIFY_DOS_INFO	566
NW_DATA_STREAM_FAT_INFO	569
NW_DATA_STREAM_SIZE_INFO	570
NW_ENTRY_INFO	571
NW_ENTRY_INFO_EXT	575
NW_ENTRY_INFO2	578
NW_IDX	583
NW_MAC_TIME	584
NW_NS_INFO	585
NW_NS_OPEN	587

NW_NS_OPENCREATE	588
NW_NS_PATH	591
SEARCH_SEQUENCE	592
20 Name Space Values	593
20.1 Access Right Values	593
20.2 Attribute Values	593
20.3 Date Values	594
20.4 Inherited Rights Mask Values	594
20.5 Name Space Flag Values	595
20.6 Basic Return Mask Values	595
20.7 Extended Return Mask Values	596
20.8 Search Attributes Values	597
20.9 Time Values	597
21 Path and Drive Concepts	599
21.1 Path Parameters	599
21.2 Network Drive Functions	600
22 Path and Drive Tasks	601
22.1 Listing Network Drives	601
22.2 Mapping Network Drives	601
22.2.1 Mapping a Network Drive Example	601
23 Path and Drive Functions	605
ConvertNameToFullPath	606
ConvertNameToVolumePath	607
NWDeleteDriveBase	608
NWGetDirBaseFromPath	610
NWGetDriveInformation	612
NWGetDriveStatus	614
NWGetDriveStatusConnRef	616
NWGetFirstDrive	618
NWGetPathFromDirectoryBase	620
NWParseNetWarePath	622
NWParsePath	624
NWSetDriveBase	627
NWSetInitDrive (obsolete 7/99)	629
NWStripServerOffPath	631
ParsePath	632
SetWildcardTranslationMode	634
StripFileServerFromPath	635
24 Server-Based Data Migration Concepts	637
24.1 Advantages of Data Migration Applications	637
24.2 Server-Based Data Migration Functions	638

25 Server-Based Data Migration Functions 639

NWGetDataMigratorInfo 640
NWGetDefaultSupportModule 641
NWGetDMFileInfo 642
NWGetDMVolumeInfo 644
NWGetSupportModuleInfo 645
NWIsDataMigrationAllowed 647
NWMoveFileFromDM 648
NWMoveFileToDM 649
NWPeekFileData 650
NWSetDefaultSupportModule 652

26 Server-Based File System Functions 653

AddSpaceRestrictionForDirectory 654
AddTrustee 656
AddUserSpaceRestriction 659
ChangeDirectoryEntry 661
DeleteTrustee 665
DeleteUserSpaceRestriction 667
GetAvailableUserDiskSpace 668
GetDiskSpaceUsedByObject 670
GetEffectiveRights 672
GetMaximumUserSpaceRestriction 675
ModifyInheritedRightsMask 677
PurgeTrusteeFromVolume 680
ReturnSpaceRestrictionForDirectory 681
ScanTrustees 683
ScanUserSpaceRestrictions 685
SetDirectoryInfo 687
UpdateDirectoryEntry 690

A Revision History 691

About This Guide

This documentation describes services that generally deal with interactions among files or functions that operate on more than one file at a time. This guide includes the following functions:

- ◆ Chapter 2, “Data Migration Functions,” on page 19
- ◆ Chapter 5, “Deleted File Functions,” on page 45
- ◆ Chapter 7, “File Engine Functions,” on page 67
- ◆ Chapter 10, “File System Functions,” on page 137
- ◆ Chapter 14, “File System Monitoring Functions,” on page 385
- ◆ Chapter 18, “Name Space Functions,” on page 439
- ◆ Chapter 23, “Path and Drive Functions,” on page 605
- ◆ Chapter 25, “Server-Based Data Migration Functions,” on page 639
- ◆ Chapter 26, “Server-Based File System Functions,” on page 653

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation.

Documentation Updates

For the most recent version of this guide, see [NLM and NetWare Libraries for C \(including CLIB and XPlat\)](http://developer.novell.com/ndk/clib.htm) (<http://developer.novell.com/ndk/clib.htm>).

Additional Information

For information about other CLib and XPlat interfaces, see the following guides:

- ◆ *NDK: NLM Development Concepts, Tools, and Functions*
- ◆ *NDK: Program Management*
- ◆ *NDK: NLM Threads Management*
- ◆ *NDK: Connection, Message, and NCP Extensions*
- ◆ *NDK: Single and Intra-File Services*
- ◆ *NDK: Volume Management*
- ◆ *NDK: Client Management*
- ◆ *NDK: Network Management*
- ◆ *NDK: Server Management*
- ◆ *NDK: Internationalization*
- ◆ *NDK: Unicode*
- ◆ *NDK: Sample Code*
- ◆ *NDK: Getting Started with NetWare Cross-Platform Libraries for C*

- ◆ *NDK: Bindery Management*

For CLib source code projects, visit [Forge \(http://forge.novell.com\)](http://forge.novell.com).

For help with CLib and XPlat problems or questions, visit the [Developer Support Forums for NLM and NetWare Libraries for C \(including CLIB and XPlat\) \(http://developer.novell.com/ndk/devforums.htm\)](http://developer.novell.com/ndk/devforums.htm). There are two for NLM development (XPlat and CLib) and one for Windows XPlat development.

Documentation Conventions

In this documentation, a greater-than symbol (>) is used to separate actions within a step and items within a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

Data Migration Concepts

1

This documentation describes Data Migration, its functions, and features.

Data Migration enables client applications to move NetWare® files to supplementary nearline storage devices. Nearline storage devices include another volume, another server, another media type, another file system, a tape or even a jukebox. Migrated files are still readily accessible, although the files themselves are remote. When the files are accessed, they are de-migrated in real time to primary storage. The files remain in the file system's directory structure and all file information stays intact.

Retrieval time for migrated files varies, depending on the nearline storage device. Retrieval from a CD ROM or disk subsystem is nearly as fast as retrieval from a NetWare volume.

Files migrated are still accessed through the NetWare file system. For example, files migrated to a jukebox remain visible in the NetWare directory and when a user attempts to access one of these files, the system retrieves the data from the jukebox.

A Data Migrator NLM application administers data migration and is available from Novell®. Support module NLM applications register with the Data Migrator to provide access to specific storage schemas. The Novell Data Migrator can register up to 32 support modules.

Users and administrators determine the criteria for migrating files. These criteria typically specify seldom accessed files or files that require excessive storage space, such as large database files. Users can migrate an unlimited number of files.

1.1 Support Module Information

All available support modules are registered with the Data Migrator under a support module ID. Call [NWGetSupportModuleInfo \(page 30\)](#) to receive a list of support modules. After receiving the IDs, use the same function to receive information about individual support modules.

The support module list is returned as a [SUPPORT_MODULE_IDS \(page 40\)](#) structure. It contains an array of support module IDs.

Information about individual modules is returned as a [SUPPORT_MODULE_INFO \(page 41\)](#) structure.

- ◆ I/O status
- ◆ Block size
- ◆ Available space
- ◆ Space in-use

Information specific to the module can also be returned as a length-preceded string.

1.2 Volume Information

NWGetDMVolumeInfo (page 27) returns information about the Data Migrator NLM on a volume. Data migration volume information includes:

- ◆ Number of migrated files
- ◆ Total size of migrated data
- ◆ Size of data on the migration media
- ◆ Amount of limbo space

Limbo space refers to migrated files that have been restored to the file system but not removed from remote storage. Generally, files are retained in remote storage after they have been migrated until the file is either deleted or re-migrated.

1.3 Data Migration Functions

These functions move files to and from remote storage, return data migration information for files and volumes, and return information about the Data Migrator and support modules.

NWMoveFileToDM	Moves a file's data to an online, long term storage media but leaves the file visible on the NetWare® volume.
NWMoveFileFromDM	Moves a file's data from an online, long term storage media to a NetWare volume.
NWGetDataMigratorInfo	Returns version numbers for the Data Migrator NLM. Use this function to test whether the Data Migrator is loaded.
NWGetDefaultSupportModule	Returns the default support module for reading and writing migrated data.
NWGetDMFileInfo	Returns information about migrated files.
NWGetDMVolumeInfo	Returns information about the data that has been migrated in relation to the specified volume.
NWGetSupportModuleInfo	Can return either a list of data migration support module IDs or information about a specific support module.
NWSetDefaultSupportModule	Sets the default support module for reading and writing migrated data.

Data Migration Functions

2

This documentation alphabetically lists the Data Migration functions and describes their purpose, syntax, parameters, and return values.

- ◆ [“NWGetDataMigratorInfo” on page 20](#)
- ◆ [“NWGetDefaultSupportModule” on page 22](#)
- ◆ [“NWGetDMFileInfo” on page 24](#)
- ◆ [“NWGetDMVolumeInfo” on page 27](#)
- ◆ [“NWGetSupportModuleInfo” on page 30](#)
- ◆ [“NWMoveFileFromDM” on page 32](#)
- ◆ [“NWMoveFileToDM” on page 34](#)
- ◆ [“NWSetDefaultSupportModule” on page 37](#)

NWGetDataMigratorInfo

Returns information about the data migrator

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT*, Windows* 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Data Migration

Syntax

```
#include <nwmigrat.h>
or
#include <nwcalls.h>

NWCCODE N_API NWGetDataMigratorInfo (
    NWCONN_HANDLE conn,
    puint32         DMPresentFlag,
    puint32         majorVersion,
    puint32         minorVersion,
    puint32         DMSMRegistered);
```

Delphi Syntax

```
uses calwin32;

Function NWGetDataMigratorInfo
  (conn : NWCONN_HANDLE;
   DMPresentFlag : puint32;
   majorVersion : puint32;
   minorVersion : puint32;
   DMSMRegistered : puint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare® server connection handle.

DMPresentFlag

(OUT) Points to a flag. If equal to -1, the DM NLM has been loaded and is running; if equal to 0, the DM NLM is not loaded.

majorVersion

(OUT) Points to the data migrator major version number.

minorVersion

(OUT) Points to the data migrator minor version number.

DMSMRegistered

(OUT) Points to a flag indicating if the support module has been registered with the data migrator: non-zero = support module was registered, zero = support module was not registered.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x897E	NCP_BOUNDARY_CHECK_FAILED
0x89FB	Data Migration is not supported

NCP Calls

0x2222 90 131 Migrator Status Info

See Also

[NWGetDMVolumeInfo \(page 27\)](#), [NWGetDMFileInfo \(page 24\)](#)

NWGetDefaultSupportModule

Returns the default read/write Support Module ID for data migration

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Data Migration

Syntax

```
#include <nwmigrat.h>
or
#include <nwcalls.h>

NWCCODE N_API NWGetDefaultSupportModule (
    NWCONN_HANDLE conn,
    puint32 supportModuleID);
```

Delphi Syntax

```
uses calwin32

Function NWGetDefaultSupportModule
  (conn : NWCONN_HANDLE;
   supportModuleID : puint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

supportModuleID

(OUT) Points to the currently supported module ID.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x00F0	ERR_INVALID_SM_ID
0x8801	INVALID_CONNECTION

0x890A	NLM_INVALID_CONNECTION
0x897E	NCP_BOUNDARY_CHECK_FAILED
0x89EC	NO_SUCH_SEGMENT
0x89FB	NO_SUCH_PROPERTY

NCP Calls

0x2222 90 134 Get/Set Default Read-Write Support Module ID

See Also

[NWSetDefaultSupportModule \(page 37\)](#), [NWGetSupportModuleInfo \(page 30\)](#)

NWGetDMFileInfo

Returns information about data migrated files

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Data Migration

Syntax

```
#include <nwmigrat.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWGetDMFileInfo (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              nameSpace,
    puint32             supportModuleID,
    puint32             restoreTime,
    puint32             dataStreams);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetDMFileInfo
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pnstr8;
   nameSpace : nuint8;
   supportModuleID : puint32;
   restoreTime : puint32;
   dataStreams : puint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired name space (optional).

path

(IN) Points to a valid path that points to a file.

nameSpace

(IN) Specifies the name space of the path (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

supportModuleID

(OUT) Points to the ID of the Support Module containing the migrated data.

restoreTime

(OUT) Points to an estimate of the time (in ticks) needed to retrieve the data.

dataStreams

(OUT) Points to an array of supported data streams.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x00F0	ERR_INVALID_SM_ID
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x897E	NCP_BOUNDARY_CHECK_FAILED
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	Bad AFP Entry ID
0x899E	INVALID_FILENAME
0x89A8	ERR_ACCESS_DENIED
0x89BF	INVALID_NAME_SPACE

Remarks

The time returned in the `restoreTime` parameter represents the estimated number of ticks needed. There are 18.2 ticks in one second.

NCP Calls

0x2222 87 06 Obtain File or Subdirectory Information

0x2222 90 129 DM File Information

See Also

[NWGetSupportModuleInfo \(page 30\)](#), [NWMoveFileFromDM \(page 32\)](#), [NWMoveFileToDM \(page 34\)](#)

NWGetDMVolumeInfo

Returns information about the Data Migrator NLM on a NetWare volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Data Migration

Syntax

```
#include <nwmigrat.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWGetDMVolumeInfo (
    NWCONN_HANDLE   conn,
    nuint16         volume,
    nuint32         supportModuleID,
    pnuint32        numberOfFilesMigrated,
    pnuint32        totalMigratedSize,
    pnuint32        spaceUsedOnDM,
    pnuint32        limboSpaceUsedOnDM,
    pnuint32        spaceMigrated,
    pnuint32        filesInLimbo);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetDMVolumeInfo
  (conn : NWCONN_HANDLE;
   volume : nuint16;
   supportModuleID : nuint32;
   numberOfFilesMigrated : pnuint32;
   totalMigratedSize : pnuint32;
   spaceUsedOnDM : pnuint32;
   limboSpaceUsedOnDM : pnuint32;
   spaceMigrated : pnuint32;
   filesInLimbo : pnuint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volume

(IN) Specifies the volume number having the migrated files.

supportModuleID

(IN) Specifies the currently supported module ID.

numberOfFilesMigrated

(OUT) Points to the migrated number of files from the selected volume.

totalMigratedSize

(OUT) Points to the total number of bytes needed to recover all the data on the selected volume.

spaceUsedOnDM

(OUT) Points to the size of the data on the migrator media.

limboSpaceUsedOnDM

(OUT) Points to the size of the demigrated data on the migrator area. Since the data is generally Read Only, the file will be kept on the migrator until the file is either deleted or remigrated with changes.

spaceMigrated

(OUT) Points to the total size of the migrated data for the volume (includes the limbo space used).

filesInLimbo

(OUT) Points to the number of files that are in limbo or were demigrated with SAVE_KEY_WHEN_FILE_IS_DEMIGRATED and have not been migrated back to the data migrator.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x00F0	ERR_INVALID_SM_ID
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8978	ERR_VOLUME_FLAG_NOT_SET
0x897E	NCP_BOUNDARY_CHECK_FAILED
0x8998	VOLUME_DOES_NOT_EXIST

NCP Calls

0x2222 90 130 Get Volume DM Status

See Also

[NWGetDefaultSupportModule](#) (page 22), [NWGetDataMigratorInfo](#) (page 20),
[NWGetSupportModuleInfo](#) (page 30)

NWGetSupportModuleInfo

Returns information about the Data Migrator NLM support modules or a list of all loaded support module IDs

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Data Migration

Syntax

```
#include <nwmigrat.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWGetSupportModuleInfo (
    NWCONN_HANDLE    conn,
    nuint32           informationLevel,
    nuint32           supportModuleID,
    puint8            returnInfo,
    puint32           returnInfoLen);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetSupportModuleInfo
  (conn : NWCONN_HANDLE;
   informationLevel : nuint32;
   supportModuleID : nuint32;
   returnInfo : puint8;
   returnInfoLen : puint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

informationLevel

(IN) Specifies the level of information to be returned. If information Level = 0, returns information about the DM NLM support module; if information Level = 1, returns a list of all loaded support module IDs.

supportModuleID

(IN) Specifies the assigned ID number of the support module migrating the data.

returnInfo

(OUT) Points to the area in which to store the information.

returnInfoLen

(OUT) Points to the size of the data area the user allocated in which to return information.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x00F0	ERR_INVALID_SM_ID
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x897E	NCP_BOUNDARY_CHECK_FAILED
0x89A8	ERR_ACCESS_DENIED
0x89FF	Failure, Invalid Info Level, or Invalid Parameter

Remarks

If the `informationLevel` parameter contains 0 (zero), the [SUPPORT_MODULE_INFO \(page 41\)](#) structure will be used to return information about the DM NLM support module to the `returnInfo` parameter. If the `informationLevel` parameter contains 1, the [SUPPORT_MODULE_IDS \(page 40\)](#) structure will be used to return a list of all loaded support module IDs to the `returnInfo` parameter.

NCP Calls

0x2222 90 132 DM Support Module Information

See Also

[NWGetDefaultSupportModule \(page 22\)](#), [NWGetDataMigratorInfo \(page 20\)](#), [NWGetDMVolumeInfo \(page 27\)](#)

NWMoveFileFromDM

Moves file data from an on-line, long term storage medium to a NetWare volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Data Migration

Syntax

```
#include <nwmigrat.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWMoveFileFromDM (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              nameSpace);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWMoveFileFromDM
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pustr8;
   nameSpace : nuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired name space (optional).

path

(IN) Points to a valid path that points to a file.

nameSpace

(IN) Specifies the name space of the path (see [Section 20.5, “Name Space Flag Values,”](#) on [page 595](#)).

Return Values

These are common return values; see [Return Values for C](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8978	ERR_VOLUME_FLAG_NOT_SET
0x897E	NCP_BOUNDARY_CHECK_FAILED
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899E	INVALID_FILENAME
0x89A8	ERR_ACCESS_DENIED
0x89FB	Invalid Namespace (abends the server)

NCP Calls

0x2222 87 06 Obtain File or Subdirectory Information

0x2222 90 133 Move File Data From DM

See Also

[NWMoveFileToDM \(page 34\)](#), [NWSetDefaultSupportModule \(page 37\)](#), [NWGetDMFileInfo \(page 24\)](#)

NWMoveFileToDM

Moves file data to an online, long term storage medium but leaves the file visible on a NetWare volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Data Migration

Syntax

```
#include <nwmigrat.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWMoveFileToDM (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              nameSpace,
    nuint32             supportModuleID,
    nuint32             saveKeyFlag);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWMoveFileToDM
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pnstr8;
   nameSpace : nuint8;
   supportModuleID : nuint32;
   saveKeyFlag : nuint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired name space (optional).

path

(IN) Points to a valid path, which points to a directory or file.

nameSpace

(IN) Specifies the name space of the path (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

supportModuleID

(IN) Specifies the assigned ID number of the support module migrating the data.

saveKeyFlag

(IN) Specifies if the migrator key will be saved when the file is demigrated:

0 Migrator key will not be saved

1 Migrator key will be saved

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x897E	NCP_BOUNDARY_CHECK_FAILED
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899E	INVALID_FILENAME
0x899C	INVALID_PATH
0x89A8	ERR_ACCESS_DENIED
0x89FB	Invalid Namespace

Remarks

If `saveKeyFlag` equals `SAVE_KEY_WHEN_FILE_IS_DEMIGRATED`, the key will be saved when the file is demigrated. This saves time because the file will not be deleted from the migrated media and will be checked for changes before subsequent migrations.

NCP Calls

0x2222 87 06 Obtain File or Subdirectory Information

0x2222 90 128 Move File Data To DM

See Also

[NWMoveFileFromDM \(page 32\)](#), [NWSetDefaultSupportModule \(page 37\)](#), [NWGetDMFileInfo \(page 24\)](#)

NWSetDefaultSupportModule

Sets the default Read/Write support module ID

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Data Migration

Syntax

```
#include <nwmigrat.h>
or
#include <nwcalls.h>

NWCCODE N_API NWSetDefaultSupportModule (
    NWCONN_HANDLE conn,
    puint32 supportModuleID);
```

Delphi Syntax

```
uses calwin32

Function NWSetDefaultSupportModule
  (conn : NWCONN_HANDLE;
   supportModuleID : puint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

supportModuleID

(IN) Points to the support module ID.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION

0x897E	NCP_BOUNDARY_CHECK_FAILED
0x89EC	NO_SUCH_SEGMENT
0x89FB	NO_SUCH_PROPERTY or INVALID_PARAMETERS

NCP Calls

0x2222 90 134 Get/Set Default Read Write Support Module ID

Data Migration Structures

3

This documentation alphabetically lists the Data Migration structures and describes their purpose, syntax, and fields.

- ◆ [“SUPPORT_MODULE_IDS” on page 40](#)
- ◆ [“SUPPORT_MODULE_INFO” on page 41](#)

SUPPORT_MODULE_IDS

Returns a list of support module IDs (level 1 information) by NWGetSupportModuleInfo

Service: Data Migration

Defined In: nwmigrat.h

Structure

```
typedef struct
{
    nuint32    numberOfSMs ;
    nuint32    SMIDs [MAX_NUM_OF_SM];
} SUPPORT_MODULE_IDS;
```

Delphi Structure

```
uses calwin32

SUPPORT_MODULE_IDS = packed Record
    numberOfSMs : nuint32;
    SMIDs : Array[0..MAX_NUM_OF_SM-1] Of nuint32
End;
```

Fields

numberOfSMs

Specifies the number of valid support module IDs returned by the Data Migrator.

SMIDs

Specifies the list of support module IDs.

SUPPORT_MODULE_INFO

Returns (level 0) support module information by NWGetSupportModuleInfo

Service: Data Migration

Defined In: nwmigrat.h

Structure

```
typedef struct
{
    nuint32    IOStatus ;
    nuint32    InfoBlockSize ;
    nuint32    AvailSpace ;
    nuint32    UsedSpace ;
    nuint8     SMInfo [MAX_SIZE_OF_SM_STRING + MAX_SIZE_OF_SM_INFO];
} SUPPORT_MODULE_INFO;
```

Delphi Structure

```
uses calwin32

SUPPORT_MODULE_INFO = packed Record
    IOStatus : nuint32;
    InfoBlockSize : nuint32;
    AvailSpace : nuint32;
    UsedSpace : nuint32; (*A length preceded string is followed by
SMInfo data*)
    SMInfo : Array[0..MAX_SIZE_OF_SM_STRING + MAX_SIZE_OF_SM_INFO - 1]
Of nuint8
End;
```

Fields

IOStatus

Specifies the IO read and write access status of the associated storage device .

InfoBlockSize

Specifies the information block size on the associated storage device.

AvailSpace

Specifies the amount of space available on the associated storage device.

UsedSpace

Specifies the amount of used space on the associated storage device. This length-preceded string is followed by `SMInfo` data.

SMInfo

Specifies the support-module specific data in the form of a length-preceded string.

Deleted File Concepts

4

This documentation describes Deleted File, its functions, and features.

NetWare® servers retain deleted files in a recoverable state. The final deallocation of a deleted file is called purging. Deleted File Services include functions for purging and recovering deleted files.

NetWare contains important changes to the file system in versions after 2.15. These changes primarily affect trustee rights, file attributes, and purgeable files.

Although differences between overlapping functions are noted, developers need to be aware of compatibility issues affecting specific functions.

4.1 Deleted File on NetWare 3.11 and above Servers

When a client erases a file on a NetWare 3.11 or above server, the server moves the file to a holding area in the directory structure of the volume. You can scan this area for deleted files by calling [NWScanForDeletedFiles \(page 54\)](#) using a search pattern. Scanning deleted files returns file information for all recoverable files in a specified directory. No prior knowledge of file names is necessary.

When you purge files on a NetWare 3.11 or above server, only the specified files are removed from the holding area. Other deleted files are not affected. Deleted files can remain on the server for an indefinite period. However, if the server must reclaim disk space, the files can be purged, after which they cannot be recovered.

4.2 Deleted File Functions

These functions handle the purging and recovery of deleted NetWare® files:

NWPurgeDeletedFile	Removes recoverable files from a NetWare server.
NWRecoverDeletedFile	Recovers deleted files from the NetWare server.
NWScanForDeletedFiles	Scans the specified directory for any deleted (salvageable) files.

Deleted File Functions

5

This documentation alphabetically lists the Deleted File functions and describes their purpose, syntax, parameters, and return values.

- ♦ [“NWPurgeDeletedFile” on page 46](#)
- ♦ [“NWRecoverDeletedFile” on page 49](#)
- ♦ [“NWRecoverDeletedFileExt” on page 52](#)
- ♦ [“NWScanForDeletedFiles” on page 54](#)
- ♦ [“NWScanForDeletedFilesExt” on page 57](#)

NWPurgeDeletedFile

Removes recoverable files from a NetWare server

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT*, Windows* 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Deleted File

Syntax

```
#include <nwdel.h>
or
#include <nwcalls.h>

NWCCODE N_API NWPurgeDeletedFile (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    nuint32             iterHandle,
    nuint32             volNum,
    nuint32             dirBase,
    const nstr8 N_FAR  *fileName);
```

Delphi Syntax

```
uses calwin32

Function NWPurgeDeletedFile
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   iterHandle : nuint32;
   volNum : nuint32;
   dirBase : nuint32;
   fileName : pnstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle to purge.

dirHandle

(IN) Specifies the directory handle for the directory containing the file to purge (valid for 3.x and above only).

iterHandle

(IN) Specifies the sequence number returned by NWScanForDeletedFiles (valid for 3.x and above only).

volNum

(IN) Specifies the volume number returned by NWScanForDeletedFiles (valid for 3.11 and above only).

dirBase

(IN) Specifies the directory base number returned by NWScanForDeletedFiles (valid for 3.11 and above only).

fileName

(IN) Points to the name of the file to purge (valid for 3.0 and 3.1 only).

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8985	NO_CREATE_DELETE_PRIVILEGES
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

For 3.x servers, only the specified file is purged.

For 3.x servers, NWPurgeDeletedFile is used in connection with NWScanForDeletedFiles. iterHandle, volNum, and dirBase are returned by NWScanForDeletedFiles and should not be modified prior to calling NWPurgeDeletedFile.

Although parameters may only be valid for some servers, each parameter must be filled. Valid parameters for NWPurgeDeletedFile on each platform are listed below:

3.0 and 3.1	3.11
conn	conn
dirHandle	dirHandle
sequence	iterHandle
	volNum
	dirBase

3.0 and 3.1

3.11

fileName

NCP Calls

0x2222 22 16 Purge Deleted File

0x2222 23 17 Get File Server Information

0x2222 87 18 Purge Salvageable File

0x2222 22 29 Purge Salvageable File

See Also

[NWScanForDeletedFiles \(page 54\)](#)

NWRecoverDeletedFile

Recovers deleted files from the NetWare server

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Deleted File

Syntax

```
#include <nwdel.h>
or
#include <nwcalls.h>

NWCCODE N_API NWRecoverDeletedFile (
    NWCONN_HANDLE    conn,
    NWDIR_HANDLE     dirHandle,
    nuint32           iterHandle,
    nuint32           volNum,
    nuint32           dirBase,
    pnstr8            delFileName,
    pnstr8            rcvrFileName);
```

Delphi Syntax

```
uses calwin32

Function NWRecoverDeletedFile
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   iterHandle : nuint32;
   volNum : nuint32;
   dirBase : nuint32;
   delFileName : pnstr8;
   rcvrFileName : pnstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle containing the deleted file.

dirHandle

(IN) Specifies the directory handle of the directory containing the file to recover.

iterHandle

(IN) Specifies the number returned by NWScanForDeletedFiles.

volNum

(IN) Specifies the number returned by NWScanForDeletedFiles.

dirBase

(IN) Specifies the number returned by NWScanForDeletedFiles.

delFileName

(OUT) Points to the name of the erased file.

rcvrFileName

(OUT) Points to the name to use in recovering the file.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x8984	NO_CREATE_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FE	File name already exists in this directory
0x89FF	Failure

Remarks

For 3.x-6.x servers, files deleted by a client are moved to a holding area on the volume until they are either purged, restored (by calling NWRecoverDeletedFile), or replaced by other deleted files.

For 3.11 servers, the recovery is performed one file at a time. NWRecoverDeletedFile can also recover the deleted file and give it a new name. This feature alleviates problems with recovering a file when a new file exists with the same name.

For 3.x, the application must specify the file name in rcvrFileName, not the path. No wildcards are allowed.

NOTE: Due to earlier support for 14 character names in NetWare, both `delFileName` and `rcvrFileName` buffers must be at least 15 bytes long.

Although parameters may only be valid for some servers, each parameter must be filled. Valid parameters for `NWRecoverDeletedFile` on each platform are listed below:

3.0 and 3.1	3.11 and above
<code>conn</code>	<code>conn</code>
<code>dirHandle</code>	<code>dirHandle</code>
<code>sequence</code>	<code>iterHandle</code>
	<code>volNum</code>
	<code>dirBase</code>
<code>deletedFileName</code> (passed in)	
<code>recoverFileName</code> (passed in)	<code>rcvrFileName</code>

NCP Calls

0x2222 22 17 Recover Erased File (old)
0x2222 22 28 Recover Salvageable File
0x2222 23 17 Get File Server Information
0x2222 87 17 Recover Salvageable File

See Also

[NWScanForDeletedFiles \(page 54\)](#)

NWRecoverDeletedFileExt

Recovers deleted files from the NetWare server, using UTF-8 strings.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Deleted File

Syntax

```
#include <nwdel.h>
or
#include <nwcalls.h>

NWCCODE N_API NWRecoverDeletedFileExt (
    NWCONN_HANDLE    conn,
    NWDIR_HANDLE     dirHandle,
    nuint32           iterHandle,
    nuint32           volNum,
    nuint32           dirBase,
    pnstr8            delFileName,
    pnstr8            rcvrFileName);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle containing the deleted file.

dirHandle

(IN) Specifies the directory handle of the directory containing the file to recover.

iterHandle

(IN) Specifies the number returned by NWScanForDeletedFilesExt.

volNum

(IN) Specifies the number returned by NWScanForDeletedFilesExt.

dirBase

(IN) Specifies the number returned by NWScanForDeletedFilesExt.

delFileName

(OUT) Points to the name of the erased file, using UTF-8 characters.

rcvrFileName

(OUT) Points to the name to use in recovering the file, using UTF-8 characters.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x8984	NO_CREATE_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FE	File name already exists in this directory
0x89FF	Failure

Remarks

Files deleted by a client are moved to a holding area on the volume until they are either purged, restored (by calling `NWRecoverDeletedFileExt`), or replaced by other deleted files.

`NWRecoverDeletedFileExt` can recover the deleted file and give it a new name. This feature alleviates problems with recovering a file when a new file exists with the same name. The application must specify the file name in `rcvrFileName`, not the path. No wildcards are allowed.

NCP Calls

0x2222 22 17 Recover Erased File (old)
0x2222 22 28 Recover Salvageable File
0x2222 23 17 Get File Server Information
0x2222 87 17 Recover Salvageable File
0x2222 89 17 Recover Salvageable File

See Also

[NWScanForDeletedFilesExt \(page 57\)](#)

NWScanForDeletedFiles

Scans the specified directory for any deleted (salvageable) files

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Deleted File

Syntax

```
#include <nwdel.h>
or
#include <nwcalls.h>

NWCCODE N_API NWScanForDeletedFiles (
    NWCONN_HANDLE          conn,
    NWDIR_HANDLE           dirHandle,
    puint32                 iterHandle,
    puint32                 volNum,
    puint32                 dirBase,
    NWDELETED_INFO N_FAR *entryInfo);
```

Delphi Syntax

```
uses calwin32;

Function NWScanForDeletedFiles
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   iterHandle : puint32;
   volNum : puint32;
   dirBase : puint32;
   Var entryInfo : NWDELETED_INFO
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the directory to scan.

iterHandle

(IN) Points to the address of the search sequence number. Must be initially set to -1.

volNum

(OUT) Points to the volume's number index (valid for 3.11 and above only).

dirBase

(OUT) Points to the directory's number index (valid for 3.11 and above only).

entryInfo

(OUT) Points to NWDELETED_INFO, containing the deleted file information.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x899B	BAD_DIRECTORY_HANDLE
0x89FF	No more salvageable files in directory

Remarks

NWScanForDeletedFiles replaces NWScanSalvageableFiles.

Initially, `iterHandle` needs to be set to -1. The server maintains the sequence number once a match has been found. No file names or wildcards are allowed in the search.

If `iterHandle` and `entryInfo` are NULL or `dirHandle` is zero, `NWScanForDeletedFiles` returns -1.

`volNum` and `dirBase` are used only when scanning NetWare 3.11 and above. These two numbers are indices used by the server to speed up the location of a deleted file. They should not be modified by an application.

Although parameters may only be valid for some servers, each parameter must be filled. The valid parameters for `NWScanForDeletedFiles` on each platform follow:

3.0 and 3.1	3.11
<code>conn</code>	<code>conn</code>
<code>dirHandle</code>	<code>dirHandle</code>
<code>sequence</code>	<code>iterHandle</code>
	<code>volNum</code>
	<code>dirBase</code>

3.0 and 3.1

3.11

entryInfo

entryInfo

NCP Calls

0x2222 22 27 Scan Salvageable Files

0x2222 23 17 Get File Server Information

0x2222 87 16 Scan Salvageable Files

See Also

[NWPurgeDeletedFile \(page 46\)](#), [NWRecoverDeletedFile \(page 49\)](#)

NWScanForDeletedFilesExt

Scans the specified directory for any deleted (salvageable) files, using UTF-8 strings.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Deleted File

Syntax

```
#include <nwdel.h>
or
#include <nwcalls.h>

NWCCODE N_API NWScanForDeletedFilesExt (
    NWCONN_HANDLE          conn,
    NWDIR_HANDLE           dirHandle,
    pnuint32                iterHandle,
    pnuint32                volNum,
    pnuint32                dirBase,
    NWDELETED_INFO_EXT     N_FAR *entryInfo);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the directory to scan. This parameter cannot be zero.

iterHandle

(IN) Points to the address of the search sequence number. Must be initially set to -1.

volNum

(OUT) Points to the volume's number index.

dirBase

(OUT) Points to the directory's number index.

entryInfo

(OUT) Points to NWDELETED_INFO_EXT, containing the deleted file information.

Return Values

These are common return values; see Return Values for C for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x899B	BAD_DIRECTORY_HANDLE
0x89FF	No more salvageable files in directory

Remarks

Initially, `iterHandle` needs to be set to -1. The server maintains the sequence number once a match has been found. No file names or wildcards are allowed in the search.

If `iterHandle` and `entryInfo` are NULL or `dirHandle` is zero, `NWScanForDeletedFilesExt` returns -1.

`volNum` and `dirBase` are indices used by the server to speed up the location of a deleted file. They should not be modified by an application.

NCP Calls

0x2222 22 27 Scan Salvageable Files
0x2222 23 17 Get File Server Information
0x2222 87 16 Scan Salvageable Files
0x2222 89 16 Scan Salvageable Files

See Also

[NWRecoverDeletedFileExt \(page 52\)](#)

Deleted File Structures

6

This documentation alphabetically lists the Deleted File structures and describes their purpose, syntax, and fields.

- ◆ [“NWDELETED_INFO” on page 60](#)
- ◆ [“NWDELETED_INFO_EXT” on page 64](#)

NWDELETED_INFO

Returns information on a deleted file

Service: Deleted File

Defined In: nwdel.h

Structure

```
typedef struct
{
    nuint32    sequence ;
    nuint32    parent ;
    nuint32    attributes ;
    nuint8     uniqueID ;
    nuint8     flags ;
    nuint8     nameSpace ;
    nuint8     nameLength ;
    nuint8     name [256];
    nuint32    creationDateAndTime ;
    nuint32    ownerID ;
    nuint32    lastArchiveDateAndTime ;
    nuint32    lastArchiverID ;
    nuint32    updateDateAndTime ;
    nuint32    updatorID ;
    nuint32    fileSize ;
    nuint8     reserved [44];
    nuint16    inheritedRightsMask ;
    nuint16    lastAccessDate ;
    nuint32    deletedTime ;
    nuint32    deletedDateAndTime ;
    nuint32    deleterID ;
    nuint8     reserved3 [16];
} NWDELETED_INFO;
```

Delphi Structure

```
uses calwin32

NWDELETED_INFO = packed Record
    sequence : nuint32;
    parent : nuint32;
    attributes : nuint32;
    uniqueID : nuint8;
    flags : nuint8;
    nameSpace : nuint8;
    nameLength : nuint8;
    name : Array[0..255] Of nuint8;
    creationDateAndTime : nuint32;
    ownerID : nuint32;
    lastArchiveDateAndTime : nuint32;
    lastArchiverID : nuint32;
```

```
updateDateAndTime : nuint32;
updatorID : nuint32;
fileSize : nuint32;
reserved : Array[0..43] Of nuint8;
inheritedRightsMask : nuint16;
lastAccessDate : nuint16;
deletedTime : nuint32;
deletedDateAndTime : nuint32;
deletorID : nuint32;
reserved3 : Array[0..15] Of nuint8
End;
```

Fields

sequence

Specifies the sequence number of the associated information.

parent

Specifies the ID of the owning subdirectory.

attributes

Specifies the attributes of the associated file.

uniqueID

Specifies the entry number of the file.

flags

Specifies the DOS attributes of the deleted file.

nameSpace

Specifies the name space of the associated file:

```
1 NW_NS_MAC
0 NW_NS_DOS
2 NW_NS_NFS
3 NW_NS_FTAM
4 NW_NS_OS2
4 NW_NS_LONG
```

nameLength

Specifies the length of the file name.

name

Specifies the file name.

creationDateAndTime

Specifies the date and time the file was created.

ownerID

Specifies the object which created the file.

lastArchiveDateAndTime

Specifies the date and time the file was last archived.

lastArchiverID

Specifies the object which last archived the file.

updateDateAndTime

Specifies the date and time the file was last updated.

updaterID

Specifies the object which last updated the file.

fileSize

Specifies the size of the file in bytes.

reserved

Is reserved for future use.

inheritedRightsMask

Specifies a bit mask of the following:

0x0000 TR_NONE
0x0001 TR_READ
0x0002 TR_WRITE
0x0004 TR_OPEN
0x0004 TR_DIRECTORY
0x0008 TR_CREATE
0x0010 TR_DELETE
0x0010 TR_ERASE
0x0020 TR_OWNERSHIP
0x0020 TR_ACCESS_CTRL
0x0040 TR_FILE_SCAN
0x0040 TR_SEARCH
0x0040 TR_FILE_ACCESS
0x0080 TR_MODIFY
0x01FB TR_ALL
0x0100 TR_SUPERVISOR
0x00FB TR_NORMAL

lastAccessDate

Specifies the date the file was last accessed.

deletedTime

Specifies the time the file was deleted.

deletedDateAndTime

Specifies the date and time the file was deleted.

deletorID

Specifies the object who deleted the file.

reserved3

Is reserved for future use.

NWDELETED_INFO_EXT

Returns information on a deleted file, using UTF-8 strings.

Service: Deleted File

Defined In: nwdel.h

Structure

```
typedef struct
{
    nuint32    sequence ;
    nuint32    parent ;
    nuint32    attributes ;
    nuint8     uniqueID ;
    nuint8     flags ;
    nuint8     nameSpace ;
    nuint8     nameLength ;
    nuint8     name [766];
    nuint32    creationDateAndTime ;
    nuint32    ownerID ;
    nuint32    lastArchiveDateAndTime ;
    nuint32    lastArchiverID ;
    nuint32    updateDateAndTime ;
    nuint32    updatorID ;
    nuint32    fileSize ;
    nuint8     reserved [44];
    nuint16    inheritedRightsMask ;
    nuint16    lastAccessDate ;
    nuint32    deletedTime ;
    nuint32    deletedDateAndTime ;
    nuint32    deleterID ;
    nuint8     reserved3 [16];
} NWDELETED_INFO_EXT;
```

Fields

sequence

Specifies the sequence number of the associated information.

parent

Specifies the ID of the owning subdirectory.

attributes

Specifies the attributes of the associated file.

uniqueID

Specifies the entry number of the file.

flags

Specifies the DOS attributes of the deleted file.

nameSpace

Specifies the name space of the associated file:

1 NW_NS_MAC
0 NW_NS_DOS
2 NW_NS_NFS
3 NW_NS_FTAM
4 NW_NS_OS2
4 NW_NS_LONG

nameLength

Specifies the length of the file name.

name

Specifies the file name, using UTF-8 characters.

creationDateAndTime

Specifies the date and time the file was created.

ownerID

Specifies the object which created the file.

lastArchiveDateAndTime

Specifies the date and time the file was last archived.

lastArchiverID

Specifies the object which last archived the file.

updateDateAndTime

Specifies the date and time the file was last updated.

updaterID

Specifies the object which last updated the file.

fileSize

Specifies the size of the file in bytes.

reserved

Is reserved for future use.

inheritedRightsMask

Specifies a bit mask of the following:

0x0000 TR_NONE
0x0001 TR_READ
0x0002 TR_WRITE
0x0004 TR_OPEN
0x0004 TR_DIRECTORY
0x0008 TR_CREATE
0x0010 TR_DELETE

0x0010 TR_ERASE
0x0020 TR_OWNERSHIP
0x0020 TR_ACCESS_CTRL
0x0040 TR_FILE_SCAN
0x0040 TR_SEARCH
0x0040 TR_FILE_ACCESS
0x0080 TR_MODIFY
0x01FB TR_ALL
0x0100 TR_SUPERVISOR
0x00FB TR_NORMAL

lastAccessDate

Specifies the date the file was last accessed.

deletedTime

Specifies the time the file was deleted.

deletedDateAndTime

Specifies the date and time the file was deleted.

deletorID

Specifies the object who deleted the file.

reserved3

Is reserved for future use.

File Engine Functions

7

This documentation alphabetically lists the File Engine functions and describes their purpose, syntax, parameters, and return values.

- ◆ [“CountComponents” on page 68](#)
- ◆ [“FEConvertDirectoryNumber” on page 70](#)
- ◆ [“FEcreat” on page 72](#)
- ◆ [“FEFlushWrite” on page 74](#)
- ◆ [“FEGetCWDnum” on page 75](#)
- ◆ [“FEGetCWVnum” on page 76](#)
- ◆ [“FEGetEntryVersion” on page 77](#)
- ◆ [“FEGetOpenFileInfo” on page 79](#)
- ◆ [“FEGetOpenFileInfoForNS” on page 82](#)
- ◆ [“FEGetOriginatingNameSpace” on page 85](#)
- ◆ [“FEMapConnsHandleToVolAndDir” on page 87](#)
- ◆ [“FEMapHandleToVolumeAndDirectory” on page 89](#)
- ◆ [“FEMapPathVolumeDirToVolumeDir” on page 90](#)
- ◆ [“FEMapVolumeAndDirectoryToPath” on page 92](#)
- ◆ [“FEMapVolumeAndDirectoryToPathForNS” on page 94](#)
- ◆ [“FEMapVolumeNumberToName” on page 96](#)
- ◆ [“FEQuickClose” on page 97](#)
- ◆ [“FEQuickFileLength” on page 99](#)
- ◆ [“FEQuickOpen” on page 101](#)
- ◆ [“FEQuickRead” on page 103](#)
- ◆ [“FEQuickWrite” on page 105](#)
- ◆ [“FERegisterNSPathParser” on page 107](#)
- ◆ [“FESetCWDnum” on page 109](#)
- ◆ [“FESetCWVandCWDnums” on page 110](#)
- ◆ [“FESetCWVnum” on page 111](#)
- ◆ [“FESetOriginatingNameSpace” on page 112](#)
- ◆ [“FEsopen” on page 114](#)

CountComponents

Returns the number of components contained in a NetWare® pathname

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileio.h>

int CountComponents (
    BYTE *pathString,
    int len);
```

Parameters

pathString

(IN) Points to the string containing the NetWare pathname.

len

(IN) Specifies the length (in bytes) of the pathString.

Return Values

This function returns the number of components in pathString.

Remarks

This function works only with NetWare path names, which can consist of a directory path, file name, and file name extension.

A NetWare path consists of a path string and a path count. The path string does not use any type of delimiter character between components of the path. Instead, the length of each path component is specified in the byte immediately preceding each component of the path string. The path count tells how many path components there are in a path. This is the number returned by CountComponents.

For example, a normal path might look like this:

```
serverName/vol2:first/second/third/file.dat
```

If serverName is assigned file server ID 1, and vol2 is assigned volume number 2, then the corresponding NetWare path format would be:

```
fileServerID = 1 volumeNumber = 2 pathString =
5first6second5third8file.dat pathCount = 4
```

The `fileServerID` and `volumeNumber` are not actually part of the `pathString`, but are kept as separate numeric values. The numbers that are part of the `pathString` are actual binary values, not their ASCII equivalents. The `pathString` is the entity that would be passed to `CountComponents` (with a length of 28, which is the total length of `pathString`), and the returned component count would be 4 (the number of component parts in `pathString`).

See Also

[_makepath \(page 149\)](#), [_splitpath \(page 318\)](#)

FEConvertDirectoryNumber

Converts a directory number in one name space to the comparable directory number in another name space

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEConvertDirectoryNumber (
    int      sourceNameSpace,
    LONG     volumeNumber,
    LONG     sourceDirectoryNumber,
    int      destinationNameSpace,
    LONG     *destinationDirectoryNumberP);
```

Parameters

sourceNameSpace

(IN) Specifies the name space of the directory number to be converted (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

volumeNumber

(IN) Specifies the volume number of the directory number to be converted.

sourceDirectoryNumber

(IN) Specifies the directory number that is to be converted.

destinationNameSpace

(IN) Specifies the name space to which the directory number is to be converted (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

destinationDirectoryNumberP

(OUT) Points to the converted directory number which corresponds to the destination name space.

Return Values

This function returns a value of 0 if successful. Otherwise, it returns a nonzero value. See Return Values for C for more information.

Remarks

A single directory entry has a different directory number for each name space that is supported on a volume. This function converts a directory number in one name space to the comparable directory number in another name space.

See Also

[FEMapHandleToVolumeAndDirectory \(page 89\)](#), [FEMapPathVolumeDirToVolumeDir \(page 90\)](#)

FEcreat

Creates a file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEcreat (
    char *name,
    int permission,
    int flagBits);
```

Parameters

name

(IN) Points to the name of the file to be opened.

permission

(IN) Specifies the file permission (if the file is being created).

flagBits

(IN) Specifies the special flags that allow more file flexibility.

Return Values

When there is no error opening the file, the function returns a file handle. When an error occurs, it returns a value of -1, and `errno` and `NetWareErrno` are set to the appropriate error codes. See Return Values for C for more information.

Remarks

This function also works on the DOS partition.

This is a special version of `creat`.

If the specified file does not exist, `FEcreat` creates the file with the specified file permission.

The permission mode is established as a combination of bits found in the `SYS\STAT.H` file. The following bits are defined:

<code>S_IWRITE</code>	The file is writeable.
-----------------------	------------------------

S_IREAD The file is readable.

A value of 0 can be specified to indicate that the file is readable and writeable.

The flag bits can be found in `nwfattr.h` and are defined as follows:

DELETE_FILE_ON_CREATE_BIT	If the file already exists, it is deleted. This allows the file to be created again.
NO_RIGHTS_CHECK_ON_OPEN_BIT	The user's rights to the file are not checked when the file is opened.
NO_RIGHTS_CHECK_ON_CREATE_BIT	The user's rights to the file are not checked when the file is created.
FILE_WRITE_THROUGH_BIT	When a file write is performed, the write function does not return until the data is actually written to the disk.
ENABLE_IO_ON_COMPRESSED_DATA_BIT	Any subsequent I/O on this entry is compressed (NetWare 4.x, 5.x, and 6.x)
LEAVE_FILE_COMPRESSED_DATA_BIT	After all I/O has been done, leave this file compressed (NetWare 4.x, 5.x, and 6.x)

See Also

[close](#)

FEFlushWrite

Flushes all pending writes for a file

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>
```

```
int FEFlushWrite (
    int handle);
```

Parameters

handle

(IN) Specifies handle of the file to be flushed.

Return Values

This function returns a value of 0 if successful. Otherwise, it returns a NetWare error code. See Return Values for C for more information.

Remarks

When this function returns, all writes associated with the file specified by the file handle are complete.

FEGetCWDnum

Returns the current working directory (CWD) number

Local Servers: nonblocking

Remote Servers: nonblocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>
```

```
LONG FEGetCWDnum (void);
```

Return Values

This function returns the CWD number (the default directory) for the current thread group.

Remarks

This function can be used by a registered path parsing function to get the CWD number when the path being parsed is a relative path.

See Also

[FESetCWDnum \(page 109\)](#), [FESetCWVandCWDnums \(page 110\)](#), [FESetCWVnum \(page 111\)](#)

FEGetCWVnum

Returns the current working volume (CWV) number

Local Servers: nonblocking

Remote Servers: nonblocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>
```

```
LONG FEGetCWVnum (void);
```

Return Values

This function returns the CWV number (the default volume) for the current thread group.

Remarks

This function can be used by a registered path parsing function to get the CWV number when the path being parsed does not include a volume name.

See Also

[FEGetCWDnum \(page 75\)](#), [FESetCWDnum \(page 109\)](#), [FESetCWVandCWDnums \(page 110\)](#), [FESetCWVnum \(page 111\)](#)

FEGetEntryVersion

Returns the version number for a directory entry (files or directories)

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>
```

```
LONG FEGetEntryVersion (
    LONG    volumeNumber,
    LONG    directoryNumber,
    BYTE    *pathString,
    LONG    pathCount,
    WORD    *version);
```

Parameters

volumeNumber

(IN) Specifies the volume number on which the entry is located.

directoryNumber

(IN) Specifies the directory number used by the directory entry.

pathString

(IN) Points to a NetWare style path string relative to the volume/directory number. This is the name of the directory entry.

pathCount

(IN) Specifies the number of elements in the path string.

version

(OUT) Points to the version number for the entry.

Return Values

See Return Values for C for more information.

0	(0x00)	Success
255	(0xFF)	Failure

Remarks

This function returns the version number for a specified directory entry. The version number of a directory entry is incremented once each time the entry is modified.

See Also

[readdir \(page 300\)](#), [stat \(page 320\)](#)

FEGetOpenFileInfo

Returns directory entry information for a given connection's file handle

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEGetOpenFileInfo (
    LONG    connection,
    LONG    handle,
    LONG    *volume,
    LONG    *directoryNumber,
    LONG    *dataStream
    LONG    *flags);
```

Parameters

connection

(IN) Specifies the connection number of the object that has the file open.

handle

(IN) Specifies the file handle for which to return `volume` or `directoryNumber`.

volume

(OUT) Points to the number of the volume on which the directory entry is located.

directoryNumber

(OUT) Points to the directory entry number of the entry.

dataStream

(OUT) Points to the data stream with which the handle is associated.

flags

(OUT) Points to the status of the handle (see Remarks section).

Return Values

See Return Values for C for more information.

0	Success
---	---------

Remarks

When given a connection number and a NetWare file handle, `FEGetOpenFileInfo` returns the information in the output parameters. The file handle for the `handle` parameter must be an OS file handle such as the `fileHandle` field returned in various FS Hooks return structures defined in `nwfshook.h`.

`FEGetOpenFileInfo` is useful if you are using FS Hooks because it gives the status/flags for an open file. However, keep in mind that `fileHandle` may not be populated by some callbacks—for example `FSHOOK_PRE_OPENFILE` if the file has not yet been opened. Also keep in mind that `FEGetOpenFileInfo` is a blocking function and cannot be used in a POST FS Hooks routine. In that case callback information would have to be passed to another routine to call `FEGetOpenFileInfo`.

The `flags` parameter is a composition of three fields from the file control block (FCB): `flags`, `extraFlags`, and `extraExtraFlags` (defined in `fileio.h`):

flags bits:

0x00000001	NotReadableBit
0x00000002	NotWritableBit
0x00000004	WrittenBit
0x00000008	DetachedBit
0x00000010	SwitchingToDirectFileSystemModeBit
0x00000020	DirectFileSystemModeBit
0x00000040	FileWriteThroughBit
0x00000080	HasFileWritePrivilegeBit

extraFlags bits:

0x00010000	DiskBlockReturnedBit
0x00020000	IAmOnTheOpenFileListBit
0x00040000	FileReadAuditBit
0x00080000	FileWriteAuditBit
0x00100000	FileCloseAuditBit
0x00200000	DontFileWriteSystemAlertBit
0x00400000	ReadAheadHintBit
0x00800000	NotifyCompressionOnCloseBit

extraExtraFlags bits:

0x01000000	IsWritingCompressedBit
0x02000000	HasTimeDateBit
0x04000000	DoingDeCompressionBit

0x08000000	NoSubAllocBit
0x10000000	IsATransactionFileBit
0x20000000	HasFileWritePrivilegeBit
0x40000000	TTSReadAuditBit
0x80000000	TTSWriteAuditBit

FEGetOpenFileInfoForNS

Returns name space specific directory entry information for a given connection's file handle

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEGetOpenFileInfoForNS (
    LONG    connection,
    LONG    handle,
    LONG    *volume,
    LONG    *DOSDirectoryNumber,
    LONG    *directoryNumber,
    LONG    *nameSpace,
    LONG    *dataStream
    LONG    *flags);
```

Parameters

connection

(IN) Specifies the connection number of the object that has the file open.

handle

(IN) Specifies the file handle for which to return `volume` or `directoryNumber`.

volume

(OUT) Points to the number of the volume on which the directory entry is located.

DOSDirectoryNumber

(OUT) Points to the DOS directory entry number of the entry.

directoryNumber

(OUT) Points to the directory entry number of the entry corresponding with `nameSpace`.

nameSpace

(OUT) Points to the name space corresponding with `directoryNumber` (see [Section 20.5, "Name Space Flag Values,"](#) on page 595).

dataStream

(OUT) Points to the data stream with which the handle is associated.

flags

(OUT) Points to the status of the handle.

Return Values

See Return Values for C for more information.

0	Success
0xFF	Failure

Remarks

When given a connection number and a NetWare file handle, FEGetOpenFileInfoForNS returns the information in the output parameters.

FEGetOpenFileInfoForNS is useful if you are using FS Hooks because it gives the status/flags for an open file as well as some name space specific directory entry information.

The `flags` parameter is a composition of three fields from the file control block (FCB): `flags`, `extraFlags`, and `extraExtraFlags` (defined in `fileio.h`):

flags bits:

0x00000001	NotReadableBit
0x00000002	NotWritableBit
0x00000004	WrittenBit
0x00000008	DetachedBit
0x00000010	SwitchingToDirectFileSystemModeBit
0x00000020	DirectFileSystemModeBit
0x00000040	FileWriteThroughBit
0x00000080	HasFileWritePrivilegeBit

extraFlags bits:

0x00010000	DiskBlockReturnedBit
0x00020000	IAmOnTheOpenFileListBit
0x00040000	FileReadAuditBit
0x00080000	FileWriteAuditBit
0x00100000	FileCloseAuditBit
0x00200000	DontFileWriteSystemAlertBit
0x00400000	ReadAheadHintBit
0x00800000	NotifyCompressionOnCloseBit

extraExtraFlags bits:

0x01000000	IsWritingCompressedBit
0x02000000	HasTimeDateBit
0x04000000	DoingDeCompressionBit
0x08000000	NoSubAllocBit
0x10000000	IsATransactionFileBit
0x20000000	HasFileWritePrivilegeBit
0x40000000	TTSReadAuditBit
0x80000000	TTSWriteAuditBit

FEGetOriginatingNameSpace

Gets the originating name space for a volume and directory number pair

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

LONG FEGetOriginatingNameSpace (
    LONG    volumeNumber,
    LONG    directoryNumber);
```

Parameters

volumeNumber

(IN) Specifies the volume number for which the originating name space is desired.

directoryNumber

(IN) Specifies the directory number for which the originating name space is desired.

Return Values

This function returns a number indicating the originating name space for the volume and directory number pair, if successful. Otherwise, it returns a value of - 1, and `errno` and `NetWareErrno` contain appropriate error codes. See Return Values for C for more information.

Remarks

This function provides useful information for file backup operations. With NetWare support for name spaces, knowing which name space created the file helps you determine the correct set of information to back up.

FEGetOriginatingNameSpace returns one of the following name spaces (LONG name space is equivalent to OS/2):

0	DOS
1	MACINTOSH
2	NFS
3	FTAM

4	LONG
5	NT

See Also

[SetCurrentNamespace \(page 444\)](#)

FEMapConnsHandleToVolAndDir

Returns a volume number and a directory number for a given connection's file handle

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEMapConnsHandleToVolAndDir (
    LONG    connectionNumber,
    int     handle,
    int     *volumeNumber,
    LONG    *directoryNumber);
```

Parameters

connectionNumber

(IN) Specifies the connection number of the object that owns the file handle.

handle

(IN) Specifies the file handle for which to return the volume and directory numbers.

volumeNumber

(OUT) Points to the number of the volume on which the directory entry is located.

directoryNumber

(OUT) Points to the directory entry number of the entry.

Return Values

See Return Values for C for more information.

0	(0x00)	Success.
255	(0xFF)	Failure.

Other NetWare errors can be returned upon failure.

Remarks

When given a connection number and a file handle, this function returns a volume number and a directory number. This information can be used to get other information about the directory entry. The file handle can be obtained from normal CLIB file I/O or from the NetWare OS.

See Also

[FEMapHandleToVolumeAndDirectory \(page 89\)](#), [FEMapVolumeAndDirectoryToPath \(page 92\)](#)

FEMapHandleToVolumeAndDirectory

Gets the volume and directory numbers being used by a file handle

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEMapHandleToVolumeAndDirectory (
    int     handle,
    int     *volumeNumberP,
    LONG    *directoryNumberP);
```

Parameters

handle

(IN) Specifies the file handle to be used to get the volume and directory numbers.

volumeNumberP

(OUT) Points to the volume number used by the file handle.

directoryNumberP

(OUT) Points to the directory number used by the file handle.

Return Values

This function returns a value of 0 if successful. Otherwise, it returns a NetWare error code. See Return Values for C for more information.

Remarks

FEMapHandleToVolumeAndDirectory returns the volume and directory numbers used by the file handle.

See Also

[FEMapPathVolumeDirToVolumeDir \(page 90\)](#), [FEMapVolumeAndDirectoryToPath \(page 92\)](#), [FEMapVolumeNumberToName \(page 96\)](#)

FEMapPathVolumeDirToVolumeDir

Maps a path consisting of a volume number, directory number, and pathname to a path consisting of a volume number and directory number

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEMapPathVolumeDirToVolumeDir (
    char    *pathName,
    int     volumeNumber,
    LONG    directoryNumber,
    int     *newVolumeNumberP,
    LONG    *newDirectoryNumberP);
```

Parameters

pathName

(IN) Points to the pathname for which the volume and directory number are desired.

volumeNumber

(IN) Specifies the volume number on which the pathname is based.

directoryNumber

(IN) Specifies the directory number on which the pathname is based.

newVolumeNumberP

(OUT) Points to the returned volume number.

newDirectoryNumberP

(OUT) Points to the returned directory number.

Return Values

This function returns a value of 0 if successful. Otherwise, it returns a NetWare error code. See Return Values for C for more information.

Remarks

If the `pathName` parameter is a full volume pathname, a new volume and directory number are returned. If the path does not include a volume, `volumeNumber` is returned for `newVolumeNumberP`. If the path is relative, `newDirectoryNumberP` is based on the directory number and pathname.

See Also

[FMapHandleToVolumeAndDirectory \(page 89\)](#), [FMapPathVolumeDirToVolumeDir \(page 90\)](#), [FMapVolumeNumberToName \(page 96\)](#)

FEMapVolumeAndDirectoryToPath

Maps a volume number and directory number to a NetWare style path

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEMapVolumeAndDirectoryToPath (
    int     volumeNumber,
    LONG    directoryNumber,
    BYTE    *pathString,
    LONG    *pathCount);
```

Parameters

volumeNumber

(IN) Specifies the volume number of the desired path.

directoryNumber

(IN) Specifies the directory number of the desired path.

pathString

(OUT) Points to the NetWare style path string.

pathCount

(OUT) Points to the path count of the returned path string.

Return Values

See Return Values for C for more information.

0	Success
0x009C	Invalid path—directory number and volume pair cannot be found
0xFFFFE	The directory number has become invalid
other NetWare errors	

Remarks

The `FEMapVolumeAndDirectoryToPath` function gets a NetWare style path (pathname and path count) from a volume number and directory number.

`FEMapVolumeAndDirectoryToPath` relies on the current name space setting of the underlying thread. If that name space does not match the name space of the volume and directory to be mapped, the function returns `0x009C`. This error can occur, for example, when the directory number comes from a file system monitoring hook, and the associated name space is something other than DOS.

To avoid the `0x009C` error, call `FEMapVolumeAndDirectoryToPath` only if the name space of the underlying thread and the name space of the directory to be mapped can be guaranteed to be identical. Otherwise, call `FEMapVolumeAndDirectoryToPathForNS`, which allows you to specify the name space. You can also call `SetCurrentNameSpace` before and after calling `FEMapVolumeAndDirectoryToPath` to set and restore the current name space of the underlying thread.

`0xFFFFE (-2)` is returned when the directory number has become invalid. This error occurs, for example, when the directory number comes from a `FSHOOK_PRE_CLOSE` file system monitoring hook, and a separate reporting procedure calls `FEMapVolumeAndDirectoryToPath` after the file has already been deleted.

See Also

[FEMapHandleToVolumeAndDirectory \(page 89\)](#), [FEMapPathVolumeDirToVolumeDir \(page 90\)](#), [FEMapVolumeNumberToName \(page 96\)](#)

FEMapVolumeAndDirectoryToPathForNS

Maps a volume number and directory number to a NetWare style path

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEMapVolumeAndDirectoryToPathForNS (
    int     volumeNumber,
    LONG    directoryNumber,
    LONG    nameSpace,
    BYTE    *pathString,
    LONG    *pathCount);
```

Parameters

volumeNumber

(IN) Specifies the volume number of the desired path.

directoryNumber

(IN) Specifies the directory number of the desired path.

nameSpace

(IN) Specifies the namespace `directoryNumber` is in (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

pathString

(OUT) Points to the NetWare-style path string.

pathCount

(OUT) Points to the path count of the returned path string.

Return Values

This function returns a value of 0 if successful. Otherwise, it returns a NetWare error code. See Return Values for C for more information.

Remarks

The `FEMapVolumeAndDirectoryToPathForNS` function is useful if you are using FS Hooks.

See Also

[FEMapHandleToVolumeAndDirectory](#) (page 89), [FEMapPathVolumeDirToVolumeDir](#) (page 90), [FEMapVolumeNumberToName](#) (page 96)

FEMapVolumeNumberToName

Maps a volume number to a volume name

Local Servers: nonblocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEMapVolumeNumberToName (
    int     volumeNumber,
    BYTE    *volumeName);
```

Parameters

volumeNumber

(IN) Specifies the volume number for which the volume name is desired.

volumeName

(OUT) Points to the name of the volume.

Return Values

Returns 0 if successful; otherwise, returns an error (see Return Values for C for more information).

Remarks

The volume name is returned as a length-preceded ASCII string.

NOTE: This function works remotely only on NetWare 3.12 and above servers.

See Also

[FEMapVolumeAndDirectoryToPath \(page 92\)](#), [NWGetVolumeName \(Volume Management\)](#)

FEQuickClose

Performs a quick close on a file on the local server

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEQuickClose (
    LONG    connection,
    LONG    task,
    LONG    fileHandle);
```

Parameters

connection

(IN) Specifies the connection closing the file.

task

(IN) Specifies task number on the connection closing the file.

fileHandle

(IN) Specifies the handle of the file to close.

Return Values

See Return Values for C for more information.

0	Success
---	---------

NetWare errors

Remarks

FEQuickClose is designated "quick" because it bypasses some of the higher I/O levels in the server libraries.

FEQuickClose is useful only in conjunction with the File System Monitoring Hooks functions and other FEQuick . . . functions. The lower level handle used with FEQuickClose is returned in FEQuickOpen and is not valid for more conventional functions like read, write, or close.

See Also

[FEQuickOpen \(page 101\)](#), [FEQuickFileLength \(page 99\)](#), [FEQuickRead \(page 103\)](#), [FEQuickWrite \(page 105\)](#), [NWAddFSMonitorHook \(page 386\)](#), [NWRemoveFSMonitorHook \(page 389\)](#)

FEQuickFileLength

Returns the length of a file opened with FEQuickOpen.

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEQuickFileLength (
    LONG    connection,
    LONG    handle,
    LONG    *fileSize);
```

Parameters

connection

(IN) Specifies the connection for opening the file.

handle

(IN) Specifies the handle of the file to check.

fileSize

(OUT) Points to the size of the file.

Return Values

See Return Values for C for more information.

0	Success
---	---------

NetWare errors

Remarks

FEQuickFileLength is designated "quick" because it bypasses some of the higher I/O levels in the server libraries.

FEQuickFileLength is useful only in conjunction with the File System Monitoring Hooks functions and other FEQuick functions.

See Also

[FEQuickClose \(page 97\)](#), [FEQuickOpen \(page 101\)](#), [FEQuickRead \(page 103\)](#), [FEQuickWrite \(page 105\)](#)

FEQuickOpen

Performs a quick open on a file on the local server

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEQuickOpen (
    LONG    connection,
    LONG    task,
    LONG    volumeNumber,
    LONG    directoryNumber,
    BYTE    *pathString,
    LONG    pathCount,
    LONG    nameSpace,
    LONG    attributeMatchBits,
    LONG    requestedAccessRights,
    LONG    dataStreamNumber,
    LONG    *fileHandle);
```

Parameters

connection

(IN) Specifies the connection opening the file.

task

(IN) Specifies the task number of the connection opening the file.

volumeNumber

(IN) Specifies the volume on which the file is located.

directoryNumber

(IN) Specifies the directory number of the file to be opened.

pathString

(IN) Points to the NetWare style path that, along with `volumeNumber`, `directoryNumber`, `pathCount`, and `nameSpace`, identifies the file to be opened.

pathCount

(IN) Specifies the number of components in `pathString`.

nameSpace

(IN) Specifies the name space in which the file resides (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

attributeMatchBits

(IN) Specifies file attributes—open the file with file attributes that match this bit mask.

requestedAccessRights

(IN) Specifies the mode of entry for opening the file (for example, read only and read/write).

dataStreamNumber

(IN) Specifies the number identifying the data stream of the file to be opened.

fileHandle

(OUT) Points to the handle that designates the open file.

Return Values

See Return Values for C for more information.

0	Success
NetWare errors	

Remarks

FEQuickOpen performs a quick open on a file specified by input parameters and returns a designating handle in `fileHandle`. FEQuickOpen is designated "quick" because it bypasses some of the higher I/O levels in the server libraries.

FEQuickOpen is useful only in conjunction with the File System Monitoring Hooks functions. The handle returned is useful only for other FEQuick ... functions.

See Also

[FEQuickClose \(page 97\)](#), [FEQuickFileLength \(page 99\)](#), [FEQuickRead \(page 103\)](#), [FEQuickWrite \(page 105\)](#) [NWAddFSMonitorHook \(page 386\)](#), [NWRemoveFSMonitorHook \(page 389\)](#)

FEQuickRead

Performs a quick read of data in a file on the local server

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>
```

```
int FEQuickRead (
    LONG    connection,
    LONG    handle,
    LONG    position,
    LONG    bytesToRead,
    LONG    *bytesRead,
    void    *buffer);
```

Parameters

connection

(IN) Specifies the connection reading the data.

handle

(IN) Specifies the handle of the file from which the data is being read.

position

(IN) Specifies the location in the file at which to start reading.

bytesToRead

(IN) Specifies the number of bytes to read.

bytesRead

(OUT) Points to number of bytes actually read.

buffer

(OUT) Points to the buffer into which the read data is stored.

Return Values

See Return Values for C for more information.

0	Success
---	---------

Remarks

FEQuickRead is designated "quick" because it bypasses some of the higher I/O levels in the server libraries.

FEQuickRead is useful only in conjunction with the File System Monitoring Hooks functions and other FEQuick functions. The lower level handle used with FEQuickRead is returned in FEQuickOpen and is not valid for more conventional functions like read, write, or close.

When FEQuickRead is successful (returns 0), the number of bytes actually read is located in the `bytesRead` parameter.

NOTE: It is the responsibility of the caller to keep track of and maintain the `position` parameter.

See Also

[FEQuickClose \(page 97\)](#), [FEQuickFileLength \(page 99\)](#), [FEQuickOpen \(page 101\)](#), [FEQuickWrite \(page 105\)](#), [NWAddFSMonitorHook \(page 386\)](#), [NWRemoveFSMonitorHook \(page 389\)](#)

FEQuickWrite

Performs a quick write of data in a file on the local server

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FEQuickWrite (
    LONG    connection,
    LONG    handle,
    LONG    position,
    LONG    bytesToWrite,
    void    *buffer);
```

Parameters

connection

(IN) Specifies the connection writing the data.

handle

(IN) Specifies the handle of the file to which the data is being written.

position

(IN) Specifies the location in the file at which to start writing.

bytesToWrite

(IN) Specifies the number of bytes to write.

buffer

(OUT) Points to the buffer into which the written data is stored.

Return Values

See Return Values for C for more information.

0	Success
---	---------

NetWare errors

Remarks

FEQuickWrite is designated "quick" because it bypasses some of the higher I/O levels in the server libraries.

FEQuickWrite is useful only in conjunction with the File System Monitoring Hooks functions and other FEQuick functions. The lower level handle used with FEQuickWrite is returned in FEQuickOpen and is not valid for more conventional functions like read, write, or close.

NOTE: It is the responsibility of the caller to keep track of and maintain the `position` parameter.

See Also

[FEQuickClose \(page 97\)](#), [FEQuickFileLength \(page 99\)](#), [FEQuickOpen \(page 101\)](#), [FEQuickRead \(page 103\)](#)

FERRegisterNSPathParser

Registers a function to convert a pathname in a name space format to the NetWare format (volume number, path, string, path count)

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

int FERRegisterNSPathParser (
    T_PathParseFunc    parser);
```

Parameters

parser

(IN) Specifies the address of a function to be called by all other functions that require a NetWare style pathname.

Return Values

Returns 0 if successful; otherwise, returns an error (see Return Values for C for more information).

Remarks

Before calling FERRegisterNSPathParser, you must set the current name space to the appropriate name space by calling SetCurrentNameSpace. Once the new path parser is registered, functions such as open call the new path parser to translate the `path` parameter into its NetWare counterparts.

To reverse FERRegisterNSPathParser, ensure that the current name space is the name space that was in effect at the time that the parsing function was registered. Then call FERRegisterNSPathParser, passing NULL for the `parser` parameter. The previously registered parser will be deleted and the default parser will be used.

When a path parse function has been registered, and conversion of a pathname to NetWare format is required by a function, the registered name space path parser is called in place of the regular NetWare API path parser.

The registered name space path parser must convert a pathname string into a NetWare pathname. A NetWare pathname consists of a path string count and a string of elements (path). The count is the number of elements that are in the path. Each element can be a length-preceded directory or filename. The NetWare path, however, does not contain the server or volume information.

The following is an example of a NetWare path. The path string count is 3; it contains three elements (dir1, dir2, and dir3).

```
\0x3dir1\0x3dir2\0x8filename
```

The prototype for the path parse function is in `nwfileng.h` and is defined as follows:

```
typedef int (*T_PathParseFunc) (  
    const char *inputPath,  
    WORD *fileServerIDp,  
    int *volumeNumberP,  
    LONG *directoryNumberP,  
    BYTE *outPathStringP,  
    LONG *outPathCountP)
```

inputPath

(IN) Input path string to be parsed.

fileServerID

(OUT) File server ID of the server where the file is located.

volumeNumberP

(OUT) Volume number of the file.

directoryNumberP

(OUT) Directory number of the file.

outPathStringP

(OUT) Path string in NetWare format.

outPathCount

(OUT) Path string count.

See Also

[SetCurrentNameSpace \(page 444\)](#)

FESetCWDnum

Sets the current working directory (CWD) number (the default directory)

Local Servers: nonblocking

Remote Servers: nonblocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>
```

```
LONG FESetCWDnum (
    LONG    CWDnum);
```

Parameters

CWDnum

(IN) Specifies the number of the directory that is to become the default directory for the current thread group.

Return Values

This function returns the old CWD number.

Remarks

The FESetCWDnum function sets the directory number that is to be used as the default for parsing pathnames that are not full pathnames.

See Also

[FEGetCWDnum \(page 75\)](#), [FEGetCWVnum \(page 76\)](#), [FESetCWVandCWDnums \(page 110\)](#), [FESetCWVnum \(page 111\)](#)

FESetCWVandCWDnums

Sets the current working volume (CWV) number and the current working directory (CWD) the default volume and directory

Local Servers: nonblocking

Remote Servers: nonblocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>
```

```
LONG FESetCWVandCWDnums (
    LONG    CWVnum,
    LONG    CWDnum) ;
```

Parameters

CWVnum

(IN) Specifies the number of the volume that is to become the default volume for the current thread group.

CWDnum

(IN) Specifies the number of the directory that is to become the default directory for the current thread group.

Return Values

This function returns the old CWD number.

Remarks

The FESetCWVandCWDnums function sets the volume and directory numbers that are to be used as the defaults for parsing pathnames that are not full volume paths.

See Also

[FESetCWDnum \(page 75\)](#), [FESetCWVnum \(page 76\)](#), [FESetCWDnum \(page 109\)](#), [FESetCWVnum \(page 111\)](#)

FESetCWVnum

Sets the current working volume (CWV) number (the default volume)

Local Servers: nonblocking

Remote Servers: nonblocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>
```

```
LONG FESetCWVnum (
    LONG    CWVnum);
```

Parameters

CWVnum

(IN) Specifies the number of the volume that is to become the default volume for the current thread group.

Return Values

This function returns the old CWV number.

Remarks

The FESetCWVnum function sets the volume number that is to be used as the default for parsing pathnames that are not full volume paths.

See Also

[FEGetCWDnum \(page 75\)](#), [FESetCWDnum \(page 109\)](#), [FESetCWVandCWDnums \(page 110\)](#)

FESetOriginatingNameSpace

Allows the user to set the originating name space of a directory entry

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>

LONG FESetOriginatingNameSpace (
    LONG    volumeNumber,
    LONG    directoryNumber,
    LONG    dirNumNameSpace,
    LONG    newNameSpace);
```

Parameters

volumeNumber

(IN) Specifies the number of the volume on which the directory entry is located.

directoryNumber

(IN) Specifies the directory number of the entry to be changed.

dirNumNameSpace

(IN) Specifies the name space number corresponding with `directoryNumber` (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

newNameSpace

(IN) Specifies the name space to be the new originating name space on the directory entry (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

Return Values

See [Return Values](#) for more information.

0	Success
-1	Fail
other NetWare errors	

Remarks

`FESetOriginatingNameSpace` returns errors on 3.x because there is no OS support.

`directoryNumber` can be an entry number for any loaded name space.

`dirNumNameSpace` specifies in which name space the `directoryNumber` is located.

FEsopen

Opens a file for shared access

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File Engine

Syntax

```
#include <nwfileng.h>
```

```
int FEsopen (
    char *name,
    int access,
    int share,
    int permission,
    int flagBits,
    BYTE dataStream);
```

Parameters

name

(IN) Points to the name of the file to be opened.

access

(IN) Specifies the access mode of the file.

share

(IN) Specifies the sharing mode of the file.

permission

(IN) Specifies the file permission (if the file is being created).

flagBits

(IN) Specifies the special flags that allow more file flexibility.

dataStream

(IN) Specifies the flag that indicates the data stream under which the file is to be opened.

Return Values

Returns a file handle upon success. Returns a value of -1, and `errno` and `NetWareErrno` are set to the appropriate error codes if errors occur. See Return Values for C for more information.

Remarks

FESopen also works on the DOS partition and is a special version of the sopen function. Call the sopen function if the primary data stream is requested rather than calling FESopen.

FESopen does not behave identically to the sopen function when only the O_CREAT and O_TRUNC bits are passed. You must also pass DELETE_FILES_ON_CREATE_BIT to the flagBits parameter in FESopen which allows the file to be deleted and created again.

The access mode is established as a combination of bits found in the FCNTL.H file and valid values follow:

O_RDONLY	The file can only be read.
O_WRONLY	The file can only be written.
O_RDWR	The file can be read or written.
O_APPEND	Records are written to the end of the file.
O_CREAT	If the file does not exist, it is created.
O_TRUNC	Any data in the file is truncated.
O_BINARY	Data is transmitted unchanged. Text mode is not supported.

The sharing mode is established as a combination of bits found in the NWSHARE.H file and valid values follow:

SH_COMPAT	Sets the compatibility mode.
SH_DENYRW	Prevents read or write access to the file.
SH_DENYWR	Prevents write access to the file.
SH_DENYRD	Prevents read access to the file.
SH_DENYNO	Permits both read and write access to the file.

NOTE: If a new file is created, the share flag is ignored.

If FESopen opens a file for compressed file I/O, the file must be opened in "exclusive mode" with SH_DENYRW. Otherwise, FESopen fails.

The permission mode is established as a combination of bits found in the SYS\STAT.H file and valid values follow:

S_IWRITE	The file is writeable.
S_IREAD	The file is readable.

A value of 0 can be specified to indicate that the file is readable and writeable.

The flag bits are in nwfattr.h and valid values follow:

DELETE_FILE_ON_CREATE_BIT	If the file already exists, it is deleted allowing the file to be created again.
NO_RIGHTS_CHECK_ON_OPEN_BIT	The rights to the file are not checked when the file is opened.
NO_RIGHTS_CHECK_ON_CREATE_BIT	The rights to the file are not checked when the file is created.
FILE_WRITE_THROUGH_BIT	When a write is performed, the write function does not return until the data is actually written to disk.
ENABLE_IO_ON_COMPRESSED_DATA_BIT	Any subsequent I/O on this entry is compressed (NetWare 4.x, 5.x, and 6.x).
LEAVE_FILE_COMPRESSED_DATA_BIT	After all I/O has been done, leave this file compressed (NetWare 4.x, 5.x, and 6.x).

NOTE: If the flag is set to `ENABLE_IO_ON_COMPRESSED_DATA_BIT` or `LEAVE_FILE_COMPRESSED_DATA_BIT` (can be ORed), the `share` parameter must be set to `SH_DENYRW` or `FEsopen` fails.

The `dataStream` parameter is a constant defined in `nwfattr.h` indicating which of the data streams (streams of data stored as separate files on the volume) associated with a file stored on a NetWare 3.x or above server is to be opened. The defined data streams are `PrimaryDataStream`, `MACResourceForkDataStream`, and `FTAMStructuringDataStream`.

See Also

`close`, `sopen` (Single and Intra-File Services)

File System Concepts

8

This documentation describes File System, its functions, and features.

File System functions enable developers to manipulate NetWare file system information. The principle operations performed by File System functions include:

- ◆ Accessing files
- ◆ Accessing directory entry information
- ◆ Managing disk space
- ◆ Monitoring file usage
- ◆ Managing trustees

You need to be aware of compatibility issues affecting specific functions. To verify a function's compatibility, see the specific reference for that function.

Functions beginning with NWInt, such as [NWIntScanFileInformation2 \(page 238\)](#), support wildcard augmentation of filename parameters. Functions ending with integers such as 2 or 3 include support for more recent file system features (such as long names).

8.1 Directory Entries

Volume Directory Entry Tables contain volume files and directories. Consequently, both files and directories are referred to as directory entries. If additional name spaces are loaded on a volume, a file or directory has a directory entry in each name space. However, DOS is the server's primary name space. Therefore, every file or directory is represented by a DOS directory entry:

- ◆ ["Directory Entry Information" on page 117](#)
- ◆ ["Directory Entry Information Access" on page 118](#)
- ◆ ["Directory Entry Attributes" on page 118](#)
- ◆ ["Directory Entry Functions" on page 119](#)
- ◆ ["Directory Information Functions" on page 119](#)

8.1.1 Directory Entry Information

The term "directory entry information" is used loosely to refer to the DOS information associated with a file or directory. The file system uses directory entry information to maintain the file or directory entry. Some of the more significant items included in directory entry information are the following:

- ◆ Short name
- ◆ Directory entry attributes
- ◆ Owner ID
- ◆ Inherited rights mask
- ◆ Entry event dates and times

Additional information is also included depending on whether the entry is a file or directory. For example, file size is returned for files and maximum space is returned for directories.

8.1.2 Directory Entry Information Access

How you access directory entry information depends on which version of NetWare® is running on the server.

NetWare 3.11 introduced multiple name space support to the NetWare file system. The set of trustee rights was modified and additional file attributes were added. The inherited rights mask now applies to files as well as directories.

See the [“Accessing File Information for 3.11 and Above” on page 132](#) task.

8.1.3 Directory Entry Attributes

Directory entry attributes are commonly known as file flags (though they also can pertain to directories). They have wide influence over the events that can or will be performed on a directory or file entry. The following table lists the attributes and explains their function:

Table 8-1 *Directory Entry Attributes*

Attribute	Bit Value	Application	Comment
A_READ_ONLY	0x00000001L	Files only.	Entry can't be written, deleted or renamed.
A_HIDDEN	0x00000002L	Files and directories.	Entry doesn't appear in a normal directory listing.
A_SYSTEM	0x00000004L	Files and directories.	Entry is used by the system and is hidden.
A_EXECUTE_ONLY	0x00000008L	Files only.	Entry can be loaded for execution only once.
A_DIRECTORY	0x00000010L	Files and directories.	Entry is a directory, not a file.
A_NEEDS_ARCHIVED	0x00000020L	Files only.	Entry has been changed since last archived.
A_SHAREABLE	0x00000080L	Files only.	Entry can be opened by multiple clients.
A_DONT_SUBALLOCATE	0x00000800L	Files only.	A file is stored in its own separately allocated memory for ease of access.
A_TRANSACTIONAL	0x00001000L	Files only.	A transaction on the entry is being tracked.
A_INDEXED	0x00002000L	Files and directories.	Not in use. Provided for compatibility only.
A_READ_AUDIT	0x00004000L	Files and directories.	Not in use.

Attribute	Bit Value	Application	Comment
A_WRITE_AUDIT	0x00008000L	Files and directories.	Not in use.
A_IMMEDIATE_PURGE	0x00010000L	Files and directories.	Entry will be purged when deleted.
A_RENAME_INHIBIT	0x00020000L	Files only.	Entry can't be renamed.
A_DELETE_INHIBIT	0x00040000L	Files and directories.	Entry can't be deleted.
A_COPY_INHIBIT	0x00080000L	Files only.	Entry can't be copied.
A_FILE_MIGRATED	0x00400000L	Files only.	Entry has been migrated.
A_DONT_MIGRATE	0x00800000L	Files only.	Entry should not be migrated.
A_IMMEDIATE_COMPRESS	0x02000000L	Files only.	Entry should be compressed when written.
A_FILE_COMPRESSED	0x04000000L	Files only.	Entry is compressed.
A_DONT_COMPRESS	0x08000000L	Files only.	Entry should not be compressed.
A_CANT_COMPRESS	0x20000000L	Files only.	Entry can't be compressed.

8.1.4 Directory Entry Functions

These functions access directory entry information. Some of these functions have older versions that are being phased out. Although both work, Novell recommends using the newer version.

NWIntMoveDirEntry	Moves or renames a directory entry (file or directory) on the same server.
NWIntScanDirectoryInformation2	Returns directory information for a directory specified by the connection handle.
NWIntScanDirEntryInfo	Obtains information about 3.x, 4.x, 5.x, and 6.x directory entries (files or directories).
NWIntScanExtendedInfo	Scans directory for the extended file information.
NWIntScanDirEntryInfo	Scans a 3.11 directory for directory entry information.
NWIntScanExtendedInfo	Scans a directory for extended directory entry information.
NWSetDirEntryInfo	Modifies information for a directory entry.
NWIntMoveDirEntry	Moves or renames a directory entry. The destination must be on the same NetWare® server.

8.1.5 Directory Information Functions

These functions are provided to access the rest of the available information for 3.11 servers and above:

NWModifyMaximumRightsMask	Modifies a directory's inherited rights mask.
NWIntScanDirectoryInformation2	Returns directory information for the specified directory.
NWSetDirectoryInformation	Changes information about the specified directory.

8.2 Directory Handles

Directory Handles identify individual directories.

A NetWare® server maintains a Directory Table for each workstation connection. This table is an array of 256 slots, each of which can point to a volume or a volume and directory path. For the DOS client, the server allocates a directory slot for each drive the workstation maps. The workstation can also request that the server enter a directory slot into the table without a drive mapping.

For each directory the NetWare server enters into the table, the server returns an index to the workstation. This value (from 1 to 256) is referred to as the directory handle. The handle provides a convenient method for referring to the associated directory.

There are several ways to acquire a directory handle for a given directory path. You can use an existing handle as is, you can modify a handle's associated path, or you can allocate a new handle. See the following tasks:

- ♦ [“Allocating a Directory Handle” on page 131](#)
- ♦ [“Accessing a Directory Handle” on page 131](#)

8.2.1 Directory Handle Functions

These functions read and manipulate directory handles. Note that many of the functions work with both regular and short directory handles.

NWAllocPermanentDirectoryHandle	Allocates a permanent directory handle and returns the caller's effective rights to the associated directory.
NWAllocTemporaryDirectoryHandle	Allocates a temporary directory handle and returns the caller's effective rights to the associated directory.
NWDeallocateDirectoryHandle	Deallocates a directory handle.
NWGetDirectoryHandlePath	Returns the path name of the directory associated with the given directory handle.
NWSetDirectoryHandlePath	Sets the path name of the directory associated with the given directory handle.

8.3 File and Directory Paths

From the client point of view, a complete NetWare file path includes the names of the NetWare server, the volume, any parent directories, and the file itself. For example, in the following file path FS1 is the server, SYS is the volume, DOC and REPORT are directories, and CHAP1.TXT is the filename:

```
FS1/SYS:DOC/REPORT/CHAP1.TXT
```


NetWare accepts forward slashes or back slashes between the components of a file path.

WARNING: All filenames and path parameters must be consistent with the name space used to access the directory entry. For DOS names, all characters should be upper case. Generally, directory handles and path names are expected to follow DOS conventions unless you are running a different OS and the corresponding name space is loaded for the specified volume.

- ♦ [“Wildcard Characters” on page 121](#)
- ♦ [“Search Attributes” on page 121](#)
- ♦ [Section 8.3.3, “UTF-8 Path and Filenames,” on page 121](#)

Also see the [“Combining a Path and Directory Handle” on page 131](#) task.

8.3.1 Wildcard Characters

Many functions accept wildcard characters within a filename parameter. For example, with [NWIntEraseFiles \(page 218\)](#) the file path can include wildcard characters, in which case a single request is able to erase multiple files. The following table shows the wildcard characters supported by NetWare®.

*	Asterisk: Zero or more characters.
?	Question mark: Any single character.

8.3.2 Search Attributes

Functions operating on directory entries typically include a search attribute. The attribute specifies the type of entries to include in the operation. The search attribute lets you include system and hidden files and files in subdirectories.

For functions that can operate on both directories and files, typically do one to the exclusion of the other. For these functions, the search attribute lets you specify whether to operate on files or directories. Below are the possible bits defined by the search attribute:

```
0x0000 SA_NORMAL
0x0002 SA_HIDDEN
0x0004 SA_SYSTEM
0x0010 SA_SUBDIR_ONLY
0x8000 SA_SUBDIR_FILES
0x8006 SA_ALL
```

8.3.3 UTF-8 Path and Filenames

NSS volumes store file and directory names in Unicode. NetWare 6.5 SP2 has added an NCP that allows you to access these names directly in UTF-8 (a Unicode encoding), rather than converting them to the server's or the client's code page. This functionality prevents the potential mangling of characters when the client and the server are using different code pages.

To use this functionality, the following requirements must be met:

- ◆ The files and directories must reside on an NSS volume.
- ◆ The server operating system must be NetWare 6.5 SP2 or later. This version adds a new set of file system NCPs: 0x2222 89.
- ◆ You must use the new file system functions and pass all path and filenames as UFT-8 strings.
- ◆ For client applications, the NetWare client must be version 4.90 SP2 or later. This version is available only for Windows 2000 and Windows XP clients

If one of these new function fails because one or more of the requirements are not met, the function converts the strings to the local code page and tries again using the old NCPs.

The following functions have been added for obtaining file system information:

- ◆ [NWAllocTempNSDirHandle2Ext \(page 453\)](#)
- ◆ [NWDeleteNSEnterExt \(page 457\)](#)
- ◆ [NWGetDirectoryBaseExt \(page 464\)](#)
- ◆ [NWGetLongNameExt \(page 470\)](#)
- ◆ [NWGetNSEnterInfoExt \(page 477\)](#)
- ◆ [NWGetNSPathExt \(page 491\)](#)
- ◆ [NWIntScanFileInformation2Ext \(page 241\)](#)
- ◆ [NWNSRenameExt \(page 505\)](#)
- ◆ [NWOpenCreateNSEnterExt \(page 510\)](#)
- ◆ [NWOpenNSEnterExt \(page 519\)](#)
- ◆ [NWReadNSInfoExt \(page 528\)](#)
- ◆ [NWScanNSEnterInfoExt \(page 536\)](#)
- ◆ [NWSetNSEnterDOSInfoExt \(page 554\)](#)
- ◆ [NWWriteNSInfoExt \(page 563\)](#)

The following functions have been added for managing trustees and effective rights:

- ◆ [NWAddTrusteeExt \(page 156\)](#)
- ◆ [NWDeleteTrusteeExt \(page 179\)](#)
- ◆ [NWGetEffectiveRightsExt \(page 203\)](#)
- ◆ [NWGetObjectEffectiveRightsExt](#) in the *Bindery Management* manual
- ◆ [NWIntScanForTrusteesExt \(page 248\)](#)
- ◆ [NWScanObjectTrusteePathsExt](#) in the *Bindery Management* manual

The following functions have been added for salvaging deleted files:

- ◆ [NWRecoverDeletedFileExt \(page 52\)](#)
- ◆ [NWScanForDeletedFilesExt \(page 57\)](#)

The following functions have been added for managing extended attributes (found in the *Single and Intra-File Services* manual):

- ◆ [NWCloseEAExt](#)

- ◆ [NWFindFirstEAExt](#)
- ◆ [NWFindNextEAExt](#)
- ◆ [NWGetEAHandleStructExt](#)
- ◆ [NWOpenEAExt](#)
- ◆ [NWReadEAExt](#)
- ◆ [NWWriteEAExt](#)

8.4 File Access

NetWare supports standard DOS services in addition to some specialized functions for accessing NetWare files. Typically, the only difference between accessing a NetWare file and a DOS file is that a NetWare file path includes server and volume names. For high level languages such as C, you can access files using the language's standard I/O functions. Similarly, in assembly language you can use the standard DOS functions.

File System Services supplement standard file IO facilities with functions that perform single-server operations. These functions can help reduce network traffic since the source and destination of the operations are contained within a single server. For NetWare 3.11 and above these functions operate on a file or a subdirectory:

- ◆ [NWFileServerFileCopy \(page 183\)](#) copies a file or a portion of a file to a new location on the same server.
- ◆ [NWRenameFile \(page 256\)](#) moves or renames a file on the same server.
- ◆ [NWIntEraseFiles \(page 218\)](#) erases NetWare system and hidden files. See [“Deleting Files” on page 133](#).

8.5 File I/O

File I/O functions provide the ability to perform the following tasks:

- ◆ Convert local file handles to NetWare® file handles
- ◆ Convert NetWare file handles to local file handles

See [“Converting File Handles” on page 132](#) for information on how to perform these tasks.

8.6 Inheritance

Rights assigned to a trustee in the parent directory apply to all subordinate directories. This is referred to as inheritance. The trustee does not need to appear in the trustee list of a subordinate directory to receive these rights.

There are a few ways to block inheritance:

- ◆ The trustee may be assigned new rights in a subordinate directory (thus overriding the inherited rights).
- ◆ The Inherited Rights Mask for the directory (or file) can be modified to include specific rights.
- ◆ The Maximum Rights Mask for the directory can be modified to exclude specific rights for all users.

When a file or directory is created, its Inherited Rights Mask includes all rights. Any rights removed from the inherited rights mask can't be inherited. An exception is the TR_SUPERVISOR bit, which can't be masked by an Inherited Rights Mask.

The Inherited Rights Mask is stored with directory entry information. See [“Directory Entry Information Access” on page 118](#) for a description of functions that read and modify this information.

The Maximum Rights Mask applies only to directories and affects all user's rights for a particular directory. While the Inherited Rights Mask is usually used to assign specific rights to a trustee, the Maximum Rights Mask is used to exclude specific rights—whether assigned or inherited—for all users in a specified directory.

8.7 Effective Rights

Effective rights take into account a directory's maximum rights and a trustee's assigned rights, inherited rights, and security equivalences to find the rights a trustee can exercise for a particular file or directory. To find the effective rights for a file or directory under your current object ID, call [NWGetEffectiveRights \(page 200\)](#).

The Maximum Rights Mask affects all user's rights for a particular directory (see [Section 8.6, “Inheritance,” on page 123](#)).

An assigned rights mask takes precedence over any inherited rights. It can remove rights that would have been inherited or grant new rights that would not have been inherited. A trustee's assigned rights are not affected by an Inherited Rights Mask. Consequently, the computation of effective rights depends on whether rights are assigned or inherited:

- ♦ If a trustee has an assigned rights mask, effective rights are computed by ORing the trustee's rights mask with any assigned rights mask of objects that the trustee is equivalent to in the bindery.
- ♦ If the trustee does not have assigned rights (either directly or through equivalence) in a given directory, the trustee inherits rights assigned (directly or through equivalence) in a superior directory. These rights are limited by the Inherited Rights Mask. The effective inherited rights are computed by ORing the trustee's inherited rights with any equivalent inherited rights, then ANDing the result with the Inherited Rights Mask.

8.8 Trustees

Directory trustees are network users assigned access rights to a directory or file. Trustees are identified by their object ID. Access rights at both the directory and files level are expressed as a bit mask.

8.8.1 Trustee Rights

The following trustee rights are defined for NetWare® 3.11 and above.

```
0x0001 TR_READ  
0x0002 TR_WRITE  
0x0004 undefined  
0x0008 TR_CREATE
```

0x0010 TR_DELETE
 0x0020 TR_ACCESS_CTRL
 0x0040 TR_FILE_SCAN
 0x0080 TR_MODIFY
 0x0100 TR_SUPERVISOR

The following table compares the privileges associated with trustee rights when assigned at the directory level and at the file level.

Table 8-2 *Directory and File Trustee Rights*

Right	Directory Level	File Level
TR_READ	Trustee can open and read the directory.	Trustee can open and read the files.
TR_WRITE	Trustee can open and write to the directory.	Trustee can open and write to the file.
TR_CREATE	Trustee can create entries in the directory.	Trustee can salvage the file after deletion.
TR_ERASE	Trustee can remove entries from the directory.	Trustee can erase the file.
TR_ACCESS_CTRL	Trustee can grant trustee rights and modify inheritance for the directory.	Trustee can grant trustee rights and modify inheritance for the file.
TR_FILE_SCAN	Trustee can scan for directory entries.	Trustee can see the file when scanning.
TR_MODIFY	Trustee can modify directory attributes and rename entries.	Trustee can modify the file's attributes (but not its content).
TR_SUPERVISOR	Trustee has all rights to the directory.	Trustee has all rights to the file.

8.8.2 Trustee Functions

These functions operate on directories or files and so are oriented more toward NetWare® 3.11 and above:

- ◆ [NWAddTrustee \(page 153\)](#)
- ◆ [NWDeleteTrustee \(page 177\)](#)
- ◆ [NWIntScanForTrustees \(page 244\)](#)

These functions operate on directories only and cannot read or set the TR_SUPERVISOR bit:

- ◆ [NWAddTrusteeToDirectory \(page 158\)](#)
- ◆ [NWDeleteTrusteeFromDirectory \(page 181\)](#)
- ◆ [NWScanDirectoryForTrustees2 \(page 262\)](#)
- ◆ [NWIntScanForTrustees \(page 244\)](#)

8.9 NLM File Information

Each network file has directory information associated with it which is stored in the server's Directory Entry Table (DET).

A file's directory information consists of the file's size, attributes, creation date, date of last access, date and time the file was last modified, and the date and time the file was last archived. It also includes the owner's object ID, object IDs of up to 6 trustees, trustee rights mask for up to 6 trustees, Inherited Rights Mask, etc:

- ♦ "File Attributes" on page 127
- ♦ "Extended File Attributes" on page 128
- ♦ "Directory Entry Table" on page 128
- ♦ "Volume Table" on page 129

The file attributes contains the information obtained by the NetWare® FLAG utility: read-only versus read/write, sharable versus nonsharable, etc.

A file's directory information can be set by calling [SetFileInfo \(page 313\)](#). In addition, [GetExtendedFileAttributes \(page 147\)](#) and [SetExtendedFileAttributes \(page 311\)](#) respectively obtain and set a part of a file's attributes called extended file attributes.

An application can call [SetFileInfo \(page 313\)](#) to set specific file information such as

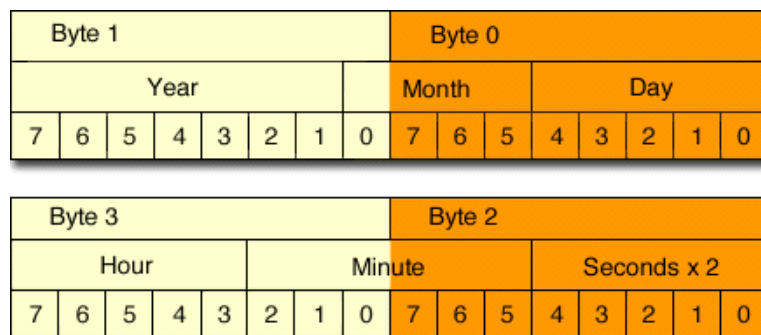
- ♦ `creationDateAndTime`—Creation date of the file (DOS format; 4 bytes)
- ♦ `fileAttributes`—File attributes to be assigned to the file
- ♦ `fileOwnerID`—Unique Bindery object ID of the file's owner (the name and Bindery object type of the file owner can be obtained by calling [NWGetObjectName \(NDK: Bindery Management\)](#)).
- ♦ `lastArchiveDateAndTime`—Last archived date and time of the file (DOS format; 4 bytes)
- ♦ `lastUpdateDateAndTime`—Last update date and time of the file (DOS format; 4 bytes).

The `creationDateAndTime`, `lastAccessDate`, `lastArchiveDateAndTime`, and `lastUpdateDateAndTime` parameters require a little interpretation. [_ConvertTimeToDOS](#) and [_ConvertDOSTimeToCalendar](#) can be used to manipulate DOS times.

- ♦ The `creationDateAndTime` parameter consists of 4 bytes indicating the hour, minute, second, year, month, and day that the file was created.
- ♦ The `lastAccessDate` parameter consists of 2 bytes indicating the year, month, and day that the file was last accessed.
- ♦ The `lastUpdateDateAndTime` and `lastArchiveDateAndTime` parameters consist of 4 bytes indicating the hour, minute, second, year, month, and day that the file was last modified or archived, respectively. The first 2 bytes of each parameter contain the year, month, and day fields, the same as the `lastAccessDate` parameter. The hour, minute, and second fields are in the second 2 bytes of each parameter:

The following figure illustrates which byte contains which element of the date and time information.

Figure 8-1 Date and Time Format



8.9.1 File Attributes

The file attributes are contained in a 4-byte field within the file's directory entry stored in the volume's DET. The attributes bytes (bytes 0 to 3) consist of flag bits whose settings can be modified.

The low-order file attribute byte contains flag bits similar to the DOS attribute byte. A client must have Modify rights to change the setting of bits in the file attribute bytes.

When set, the bits in the low-order attribute byte (byte 0) have the following meanings:

- 0 Read Only Bit
- 1 Hidden Bit
- 2 System Bit
- 3 Execute Only Bit
- 4 Subdirectory Bit
- 5 Archive Bit
- 6 Undefined
- 7 Share Bit

The following table gives the attribute bits that are set for each possible mode setting (the `_A` constants are defined in `DIRECT.H`):

Mode	Attributes			
None	<code>_A_EXECUTE</code>	<code>_A_NODELET</code>	<code>_A_NORENAM</code>	<code>_A_SYSTEM</code>
R	<code>_A_RDONLY</code>	<code>_A_NODELET</code>	<code>_A_NORENAM</code>	
W	<code>_A_HIDDEN</code>	<code>_A_NODELET</code>	<code>_A_NORENAM</code>	
X	<code>_A_EXECUTE</code>			
RW	None			
RX	<code>_A_RDONLY</code>	<code>_A_NODELET</code>	<code>_A_NORENAM</code>	
WX	<code>_A_HIDDEN</code>	<code>_A_NODELET</code>	<code>_A_NORENAM</code>	
RWX	None			

The access and chmod functions indirectly work on the attributes in byte 0. The attribute bits in this byte are used to emulate what is called the mode of the file under UNIX.

8.9.2 Extended File Attributes

The [GetExtendedFileAttributes \(page 147\)](#) and [SetExtendedFileAttributes \(page 311\)](#) functions obtain and set the second file attribute byte (byte 1) by passing a file path and extended file attributes byte.

The bits in byte 1 have the following meanings:

- 3 Don't suballocate bit (set this bit to disallow suballocation on this entry)
- 4 Transaction bit (used by TTS)
- 6 Read audit bit (unused)
- 7 Write audit bit (unused)

The Index file attribute is no longer supported since all the files are automatically indexed when they have 64 or more regular File Allocation Table (FAT) entries and are randomly accessed.

The following bits are defined for byte 2:

- 0 Immediate purge bit
- 1 Rename inhibit bit
- 2 Delete inhibit bit
- 3 Copy inhibit bit
- 7 Data migration inhibit bit

NetWare 4.x, 5.x, and 6.x also define the following attributes in byte 3:

- 0 Data save key (used for data migration)
- 1 Immediately compress file (or all files in subdirectory)
- 2 Data stream compressed
- 3 Do not compress this entry
- 4 Create a hard link entry (for NFS)
- 5 Cannot compress data stream
- 6 Attribute archive bit

8.9.3 Directory Entry Table

To record information about directories and files, a server maintains a Directory Entry Table (DET). The DET consists of several types of 128-byte entries, including directory nodes, file nodes, and trustee nodes.

A directory node includes the following information about a directory: directory name, attributes, inherited rights mask, creation date and time, creator's object ID, a link to the parent directory, and a link to a trustee node (if one exists). It also includes a name space indicator, last archived date and time, last modification date and time, up to 8 trustee object IDs, up to 8 trustee rights masks.

A file node includes the following information about a file: filename, attributes, file size, creation date and time, deletion date and time, owner's object ID, object ID of the object that performed the last deletion, object IDs of up to 6 trustees, trustee rights mask for up to 6 trustees, inherited rights mask, last-accessed date, last-updated date and time, and a link to a directory.

A trustee node includes the following information: the object IDs of 2 to 16 trustees of a directory linked to the trustee node, 2 to 16 corresponding trustee rights masks, a link to a directory, and a link to the next trustee node (if one exists).

8.9.4 Volume Table

To record information about volumes, a server maintains a Volume Table that includes the number of volumes mounted in the server, the name, size, and other information pertaining to each volume. Functions that return information about volumes access the Volume Table.

8.10 Directory Task Functions

These functions create, delete, and rename directories:

NWCreateDirectory	Creates a NetWare® directory on the specified NetWare server.
NWDeleteDirectory	Deletes a NetWare directory.
NWRenameDirectory	Renames a NetWare directory.

8.11 Directory Space Functions

These functions access directory space limits and return directory space information:

NWGetDirSpaceLimitList	Returns the actual space limitations for a directory.
NWGetDirSpaceInfo	Returns directory space information.
NWSetDirSpaceLimit	Limits the space available on a specified directory.

8.12 File Handle Conversion Functions

These functions provide the ability to convert between local and NetWare® file handles:

NWConvertFileHandle	Converts a local file handle to a NetWare file handle.
NWConvertHandle	Converts a NetWare file handle to a local file handle.

8.13 File Information Functions

These functions search for files, access file information, and monitor file usage. Some of these functions have older versions that are being phased out. Although both work, Novell® recommends using the newer version.

NWGetSparseFileBitMap	Returns a bit map showing which blocks in a sparse file contain data.
NWIntFileSearchContinue	Performs a search operation for files on the specified volume.
NWIntFileSearchInitialize	Initializes a search operation for files on the specified volume.
NWGetExtendedFileAttributes2	Returns the extended attributes for the specified file.

NWGetFileConnectionID	Returns the connection ID of the NetWare server that owns the specified file handle.
NWIntScanFileInformation2	Scans the specified directory for the specified file and returns the file's directory entry information.
NWSetCompressedFileSize	Attempts to set the logical file size for a compressed file.
NWSetExtendedFileAttributes2	Modifies the extended attributes for the specified file.
NWSetFileAttributes	Modifies the attributes for the specified file.
NWSetFileInformation2	Modifies file information for the specified file.

8.14 File Task Functions

These functions erase, copy, and rename files on a NetWare® server. Some of these functions have older versions that are being phased out. Although both work, Novell® recommends using the newer version.

NWIntEraseFiles	Deletes NetWare files from a server.
NWFileServerFileCopy	Copies from one file to another. The source and target directories must be on the same NetWare server.
NWIntEraseFiles	Deletes NetWare files from the server.
NWIntFileSearchContinue	Iteratively retrieves all directory entries matching <code>searchPath</code> .
NWRenameFile	Moves or renames a file.

8.15 File Usage Functions

These functions return file usage statistics:

NWScanConnectionsUsingFile	Returns a list of workstation connection numbers for connections using the specified file.
NWScanOpenFilesByConn2	Returns information for files currently opened by the specified connection.

This documentation describes common tasks associated with File System.

9.1 Directory-Based Tasks

These tasks help access and manage a directory:

- ♦ [“Allocating a Directory Handle” on page 131](#)
- ♦ [“Accessing a Directory Handle” on page 131](#)
- ♦ [“Combining a Path and Directory Handle” on page 131](#)
- ♦ [“Accessing File Information for 3.11 and Above” on page 132](#)

9.1.1 Allocating a Directory Handle

Directory handles can be permanent or temporary. A temporary handle is deleted as soon as the process that allocated the handle terminates. Permanent handles persist until the connection is closed or a process specifically deallocates them.

Separate functions allocate temporary and permanent directory handles:

- ♦ [NWAllocPermanentDirectoryHandle \(page 161\)](#)
- ♦ [NWAllocTemporaryDirectoryHandle \(page 163\)](#)

Call [NWDeallocateDirectoryHandle \(page 173\)](#) to deallocate a directory handle. It is especially important to deallocate permanent handles since they can remain after your application terminates.

9.1.2 Accessing a Directory Handle

A pair of functions read and modify the file path associated with a directory handle:

- ♦ [NWGetDirectoryHandlePath \(page 191\)](#)
- ♦ [NWSetDirectoryHandlePath \(page 271\)](#)

9.1.3 Combining a Path and Directory Handle

Many functions allow you to combine a path with a directory handle to specify a file or directory. If the directory handle parameter is a nonzero value, these functions generally interpret the path relative to the directory associated with the handle. Including a directory handle with a file operation can reduce the amount of space required to store the path variable.

9.1.4 Accessing File Information for 3.11 and Above

File System Services provide access to directory entry information in the DOS name space. (Name Space Services provide access to entry information in other name spaces.) A pair of functions read and set directory entry information:

- ◆ [NWIntScanDirEntryInfo \(page 232\)](#) reads directory entry information.
- ◆ [NWSetDirEntryInfo \(page 277\)](#) modifies directory entry information.

These functions operate on NetWare® 3.11 and above only. They use three structures to pass directory entry information across the NetWare interface: [NWENTRY_INFO \(page 353\)](#), [NWFILE_INFO \(page 357\)](#), and [NWDIR_INFO \(page 351\)](#).

The following code calls [NWSetDirEntryInfo \(page 277\)](#) to return some information about either a file or a directory. The command line supplies the directory path and search string, and also indicates whether to scan for files or directories. For example, if you want file directory entry information and PROG is the name of the executable, FS1 is the server, DIR1 is the directory, and *.* is the search string for files, the command line would be:

```
PROG FS1:\DIR1 *.*
```

If you want directory information the command line would be:

```
PROG FS1:\DIR1 * /d
```

[NWParseNetWarePath \(page 622\)](#) finds the connection handle, and [NWAllocTemporaryDirectoryHandle \(page 163\)](#) gets a directory handle to the input path. [NWSetDirEntryInfo \(page 277\)](#) is then called until it returns an error. Results are displayed for each entry found. The inherited rights mask is shown for directories, and the file attributes are shown for files.

9.2 File-Based Tasks

These tasks help you manage NetWare files:

- ◆ [“Locating Files” on page 132](#)
- ◆ [“Converting File Handles” on page 132](#)
- ◆ [“Deleting Files” on page 133](#)

9.2.1 Locating Files

The beginning and ending of NetWare® files can be located using lseek found with most C compilers.

9.2.2 Converting File Handles

The two basic types of file handles generated in the network environment are local file handles and NetWare® file handles. Local file handles are created and accessed by the local OS running on an individual workstation. NetWare file handles are created for files on the network and are accessed by the NetWare OS. Two functions convert these two types of file handles from one form to the other:

- ◆ [NWConvertFileHandle \(page 166\)](#)
- ◆ [NWConvertHandle \(page 168\)](#)

[NWConvertFileHandle \(page 166\)](#) converts a file handle allocated by a local OS to a four-byte or six-byte NetWare file handle. Along with returning the NetWare handle, this function also returns the references of the connection containing the NetWare handle. [NWConvertFileHandle \(page 166\)](#) does not create a NetWare file handle, rather it returns an existing NetWare handle. Therefore the function will fail if the local file handle is not associated with a NetWare file.

[NWConvertHandle \(page 168\)](#) creates a local file handle from a NetWare file. This function should be called only once per file because it creates a new local file handle and allocates resources each time it is called. The local file handle should be closed using the local OS's close file call.

9.2.3 Deleting Files

NetWare® files can be deleted on a server using [NWIntEraseFiles \(page 218\)](#).

9.3 Disk Space Management Tasks

With NetWare® 3.11 and above, you can control the total amount of space available within a directory and monitor usage for each connection.

9.3.1 Limiting Directory Space

NetWare® 3.11, 3.12, 4.x, 5.x, and 6.x servers let you restrict the amount of space allocated to a directory. Directory space limits are specified in 4K blocks. A pair of functions read and set directory space limits:

- ♦ [NWGetDirSpaceLimitList \(page 195\)](#) returns the space limit for a directory.
- ♦ [NWSetDirSpaceLimit \(page 281\)](#) sets a directory's space limit.

9.3.2 Monitoring File Usage

File System includes two functions that monitor file usage on a connection basis:

- ♦ [NWScanConnectionsUsingFile \(page 260\)](#) scans for a list of connections using a specified file. It returns [CONNS_USING_FILE \(page 332\)](#) to give the various counts for the file, such as the use count and the open count. For each connection accessing the file, the task number, lock status, and access control are also included.
- ♦ [NWScanOpenFilesByConn2 \(page 265\)](#) scans for a list of files opened by a specified connection. It returns an [OPEN_FILE_CONN \(page 359\)](#) structure identifying the file, and includes information such as the lock status and access control.

These functions are compatible with NetWare® 3.x and above although there are some differences in the information returned across versions.

9.4 Trustee Tasks

Duplicate functions exist for adding, deleting, and scanning trustees. One group of functions operates both on directories and files; the other operates only on directories.

9.4.1 Adding and Deleting File System Trustees

To add to or delete from a file or directory's trustee list, you supply the path specification and a trustee object ID. When adding a trustee you also specify the trustee's rights mask. Only static objects can be added as trustees. If the added object is a trustee already, the trustee's current rights mask is replaced by the new one.

9.4.2 Scanning File System Trustees

You can scan for trustees across multiple directories. When you scan for trustees, trustee information is returned as an array of `TRUSTEE_INFO` (page 371). (`NWIntScanForTrustees` (page 244) nests this structure within `NWET_INFO` (page 355).) Information for up to 20 trustees can be returned per iteration.

9.5 NLM-Based Tasks

These two tasks assist you in managing file systems with NLMs:

- ◆ “[Accessing Files on a Server \(NLM\)](#)” on page 134
- ◆ “[Purging and Salvaging Files \(NLM\)](#)” on page 135

9.5.1 Accessing Files on a Server (NLM)

Most NetWare® File functions identify files by a file path. The file path can be an absolute with a volume name or it can be relative to the current working directory (CWD):

- ◆ Absolute Path—Specify the entire path to the target directory or file as the `pathName` parameter.
- ◆ Relative Path—Specify a current working directory (CWD) using `chdir`. Then specify a directory or file path as the `pathName` parameter. The full path to the target directory or file is the concatenation of the CWD parameter followed by the `pathName` parameter.

File Services functions do not require a server name as a parameter. The target server is always the server to which the NLM™ application is currently logged in (or connected in the case of the local server).

File paths can be up to 255 bytes and must be NULL-terminated. When specifying a file to a File Services function, format the file path as follows:

```
volume:directory\...\directory\filename
```

The volume name can be up to 16 characters long and must include a terminating colon (:). The name cannot include spaces or the following characters:

*	Asterisk
?	Question mark
:	Colon
\	Backslash
/	Slash

Filenames and directory names on the network are represented as strings with periods embedded as normal characters. Filenames and directory names can be from 1 to 8 characters and can include a 1 to 3 character extension.

Some NetWare File functions accept wildcard characters in filenames. NetWare supports a larger set of wildcard characters than does DOS.

The following wildcard characters can be used:

* An asterisk matches zero or more characters. The pattern * therefore matches any string without an extension. The pattern *.* matches anything.

The network wildcard substitution algorithm is implemented as follows:

- ♦ All characters except the wildcard characters are treated as normal characters.
- ♦ In a search pattern, the wildcard characters must match the characters recorded in the file and directory names on the network.

9.5.2 Purging and Salvaging Files (NLM)

An application can mark files for deletion with [remove \(page 302\)](#) or [unlink \(page 325\)](#). These functions cause files to be marked for deletion. A file marked for deletion is not automatically erased until another file needs the space it occupies. The NetWare® 3.x and above OS saves deleted files (and all information about those files) in their original directory until the server runs out of disk allocation blocks on the volume or until the files marked for deletion are purged.

The [SalvageErasedFile \(page 307\)](#) function can be used to salvage a file that has been marked for deletion. The [PurgeErasedFile \(page 298\)](#) function can be used to permanently delete a file marked for deletion. Files deleted with [PurgeErasedFile \(page 298\)](#) cannot be recovered.

See [Salvaging Files: Example \(NDK: Sample Code\)](#).

File System Functions

10

This documentation alphabetically lists the File System functions and describes their purpose, syntax, parameters, and return values.

10.1 A*-M* Functions

Click on any function name in the table of contents to view the purpose, syntax, parameters, and return values for that function.

- ♦ [“access” on page 138](#)
- ♦ [“chdir” on page 140](#)
- ♦ [“chmod” on page 141](#)
- ♦ [“closedir” on page 143](#)
- ♦ [“FileServerFileCopy” on page 144](#)
- ♦ [“getcwd” on page 146](#)
- ♦ [“GetExtendedFileAttributes” on page 147](#)
- ♦ [“_makepath” on page 149](#)
- ♦ [“mkdir” on page 151](#)

access

Determines whether a file or directory exists and if it can be accessed

Local Servers: blocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <unistd.h>

int access (
    const char *path,
    int mode);
```

Parameters

path

(IN) Specifies the string containing the path that includes the file or directory to be accessed (maximum 255 characters, including the NULL terminator).

mode

(IN) Specifies the access permission mode for the file.

Return Values

Returns 0 if the file or directory exists and can be accessed with the specified mode. Otherwise, it returns a value of -1. If an error occurs, the `errno` parameter is set.

Remarks

`access` also works on the DOS partition.

`access` determines if the file or directory specified by the `path` parameter exists and if it can be accessed with the file permission given by the `mode` parameter.

When the `mode` parameter is 0, only the existence of the file is verified. The read and/or write and/or execute permission for the file can be determined when the bits of the `mode` parameter are a combination of the following:

0	F_OK: File existence
1	X_OK: Execute permission
2	W_OK: Write permission

4 R_OK: Read permission

The result is dependent on the current connection number.

The SetCurrentNameSpace function sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare® versions before 4.x, access works with only the DOS name space for remote servers.

See [Using access\(\): Example \(NDK: Sample Code\)](#).

See Also

[chmod \(page 141\)](#), [fstat](#) (Single and Intra-File Services)

chdir

Changes the current working directory to the specified path name

Local Servers: blocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <unistd.h>

int chdir (
    const char *pathname);
```

Parameters

pathname

(IN) Specifies the buffer containing the directory path (can include a volume name).

Return Values

Returns a value of 0 if successful, nonzero otherwise. If an error occurs, `errno` and `NetWareErrno` are set.

Remarks

`chdir` causes all threads in the current thread group to have a new current working directory. The `pathname` parameter can be either relative to the current working directory or it can be an absolute path name.

The `SetCurrentNameSpace` function sets the name space which is used for parsing the path input to `chdir`.

NOTE: For NetWare versions before 4.x, `chdir` works with only the DOS name space for remote servers.

See Also

[getcwd \(page 146\)](#), [mkdir \(page 151\)](#), [rmdir \(page 306\)](#)

chmod

Changes the file access mode

Local Servers: blocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <stat.h>
```

```
int chmod (
    const char *path,
    int mode);
```

Parameters

path

(IN) Specifies the string containing the path that includes the file whose access mode is to be modified (maximum 255 characters, including the NULL terminator).

mode

(IN) Specifies the access permission mode for the file.

Return Values

Returns a value of 0 if successful, -1 otherwise. If an error occurs, `errno` is set.

Remarks

To call `chmod`, you must meet the following requirements:

- ◆ The current connection must have modify permission to the specified file.
- ◆ The target namespace must be DOS. To set the target namespace, use `SetTargetNameSpace(NW_NS_DOS)`.
- ◆ For remote servers, the current name space must be DOS on NetWare versions before 4.x. See `SetCurrentNameSpace`.

`chmod` works on all NetWare file systems, including the DOS partition.

The various mode settings are given in the `SYS\STAT.H` header file. The access permissions for the file are specified as a combination of bits defined in the `SYS\STAT.H` header file.

<code>S_IWRITE</code>	The file is writable
-----------------------	----------------------

S_IREAD	The file is readable
---------	----------------------

Alternatively, zero can be specified to indicate that the file is readable and writable.

See Also

[fstat](#) (Single and Intra-File Services), [SetCurrentNamespace](#) (page 444), [SetTargetNamespace](#) (page 446), [stat](#) (page 320)

closedir

Closes a specified directory

Local Servers: nonblocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <dirent.h>
```

```
int closedir (  
    DIR *dirP);
```

Parameters

dirP

Specifies the directory to be closed.

Return Values

0x00	ESUCCESS
0x04	EBADF
NetWare Error	UNSUCCESSFUL

Remarks

closedir closes the directory specified by the `dirP` parameter and frees the memory allocated by the `opendir` function. All open directories are automatically closed when an NLM™ application is terminated.

See Also

[opendir \(page 296\)](#), [readdir \(page 300\)](#)

FileServerFileCopy

Copies a file, or a portion of a file, to another file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfinfo.h>

int FileServerFileCopy (
    int     sourceFileHandle,
    int     destinationFileHandle,
    LONG    sourceFileOffset,
    LONG    destinationFileOffset,
    LONG    numberOfBytesToCopy,
    LONG    *numberOfBytesCopied);
```

Parameters

sourceFileHandle

(IN) Specifies the file handle of the source file.

destinationFileHandle

(IN) Specifies the file handle of the destination file.

sourceFileOffset

(IN) Specifies the offset (in bytes) in the source file where copy should begin.

destinationFileOffset

(IN) Specifies the offset (in bytes) in the destination file where the data should be copied.

numberOfBytesToCopy

(IN) Specifies the number of bytes to be copied.

numberOfBytesCopied

(OUT) Points to the number of bytes actually copied.

Return Value

0	0x00	ESUCCESS
1	0x01	ERR_INSUFFICIENT_SPACE

22	0x16	EBADHNDL
131	0x83	ERR_NETWORK_DISK_IO
136	0x88	ERR_INVALID_FILE_HANDLE
147	0x93	ERR_NO_READ_PRIVILEGE
148	0x94	ERR_NO_WRITE_PRIVILEGE_OR_READONLY
149	0x95	ERR_FILE_DETACHED
162	0xA3	ERR_IO_LOCKED

Remarks

An application must pass file handles in the `sourceFileHandle` and `destinationFileHandle` parameters. A file handle can be obtained by calling the `open`, `sopen`, `creat`, or `fileno` function.

To copy from the beginning of the source file to a new file, set the `sourceFileOffset` and `destinationFileOffset` parameters to `0x00`.

To copy the entire source file, specify a value in the `numberOfBytesToCopy` parameter that matches or exceeds the file size.

The `numberOfBytesCopied` parameter returns the number of bytes copied between files as a result of calling this function.

See Also

`creat`, `fileno`, `open`, `sopen` (Single and Intra-File Services)

getcwd

Returns the current working directory of the current thread group

Local Servers: either blocking or nonblocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <unistd.h>
```

```
char *getcwd (
    char    *buffer,
    size_t  size);
```

Parameters

buffer

(OUT) Specifies the buffer in which to place the current working directory.

size

(IN) Specifies the length of buffer (including space for the delimiting \0 character).

Return Values

Returns the address of the string containing the name of the current working directory if successful. Otherwise, NULL is returned and `errno` is set.

Remarks

When the `buffer` parameter is NULL, a string is allocated to contain the current working directory. This string must be freed (by calling the `free` function) or NetWare will issue a leaked memory error at unload time.

Blocking Information Locally, `getcwd` blocks when the `buffer` parameter is NULL and does not block when the `buffer` parameter is not NULL.

See Also

[chdir \(page 140\)](#), [free](#), [mkdir \(page 151\)](#), [rmdir \(page 306\)](#)

GetExtendedFileAttributes

Returns the extended attributes for a file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfileio.h>

int GetExtendedFileAttributes (
    char *filePath,
    BYTE *extendedFileAttributes);
```

Parameters

filePath

(IN) Points to a string containing the absolute path or path relative to the current working directory of the file for which to get extended file attributes (maximum 255 characters, including the NULL terminator).

extendedFileAttributes

(OUT) Points to the extended attributes.

Return Value

0	0x00	ESUCCESS
137	0x89	ERR_NO_SEARCH_PRIVILEGE
156	0x9C	ERR_INVALID_PATH
158	0x9F	ERR_BAD_FILE_NAME
191	0xBF	ERR_INVALID_NAME_SPACE
253	0xFD	ERR_BAD_STATION_NUMBER
254	0xFE	ERR_SPOOL_DIRECTORY_ERROR
255	0xFF	ERR_NO_FILES_FOUND—The target file does not exist.

Remarks

GetExtendedFileAttributes returns the value of the first byte of the file attributes, known as the extended attributes byte. The following bits are defined:

- 3 Don't Suballocate (set this bit to disallow suballocation on this entry)
- 4 Transaction (used by TTS)
- 6 Read Audit (unused)
- 7 Write Audit (unused)

NOTE: Do not confuse the file attributes byte with true extended attributes, which can be manipulated with the Extended Attribute functions.

If the transaction bit is set in the `extendedFileAttributes` parameter, NetWare TTS™ software tracks all writes to the file during a transaction. A transaction file cannot be deleted or renamed until the transaction bit is turned off with the `SetExtendedFileAttributes` function.

An application can specify a file in several ways. For example, suppose the full path of the file `TARGET.DAT` is:

```
SYS:ACCOUNT\DOMEST\TARGET.DAT
```

and the current working directory is `SYS:ACCOUNT`. The application can specify the partial path, `DOMEST\TARGET.DAT`, or the full path in the `filePath` parameter.

`GetExtendedFileAttributes` requires that the current connection have See File rights to the directory where the file resides.

The `SetCurrentNameSpace` function sets the name space which is used for parsing the path input to `GetExtendedFileAttributes`.

NOTE: For NetWare versions before 4.x, `GetExtendedFileAttributes` works with only the DOS name space for remote servers.

See Also

[SetExtendedFileAttributes \(page 311\)](#)

_makepath

Constructs a full NetWare path name

Local Servers: blocking

Remote Servers: N/A

Platform: NLM

Service: File System

Syntax

```
#include <nwfileio.h>
```

```
void _makepath (
    char      *path,
    const char *volume,
    const char *dir,
    const char *fname,
    const char *ext);
```

Parameters

path

(OUT) Points to the string containing the full path name.

volume

(IN) Specifies the volume name.

dir

(IN) Specifies the directory name.

fname

(IN) Specifies the base name of the file without an extension.

ext

(IN) Specifies the file name extension.

Remarks

The NetWare path name is constructed from the components consisting of a volume name, directory path, file name, and file name extension. The full path name is placed in the buffer pointed to by the `path` parameter.

The maximum size required for each buffer is specified by the manifest constants which are defined in the `NWDIR.H` file.

```
255  _MAX_PATH
 16  _MAX_VOLUME (volume name length)
255  _MAX_DIR
```

9 `_MAX_FNAME`
5 `_MAX_EXT`

See [Using `_makepath` and `_splitpath`: Example](#) (*NDK: Sample Code*).

See Also

[_splitpath](#) (page 318)

mkdir

Creates a new directory with a specified mode

Local Servers: blocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <stat.h>

int mkdir (
    const char *path);
```

Parameters

path

(IN) Points to the path containing the new directory (either relative to the current working directory or an absolute path name).

Return Values

Returns a value of 0 if successful, nonzero otherwise.

Remarks

mkdir also works on the DOS partition.

The current connection must have Create rights in the parent directory. The inherited rights mask for the new directory is ALL rights.

The SetCurrentNameSpace function sets the name space used for parsing the path input to mkdir.

For NetWare versions before 4.x, mkdir works with only the DOS name space for remote servers.

See Also

[chdir \(page 140\)](#), [getcwd \(page 146\)](#), [rmdir \(page 306\)](#)

10.2 NWA*-NWF* Functions

Click on any function name in the table of contents to view the purpose, syntax, parameters, and return values for that function.

- ◆ [“NWAddTrustee” on page 153](#)

- ◆ “NWAddTrusteeExt” on page 156
- ◆ “NWAddTrusteeToDirectory” on page 158
- ◆ “NWAllocPermanentDirectoryHandle” on page 161
- ◆ “NWAllocTemporaryDirectoryHandle” on page 163
- ◆ “NWConvertFileHandle” on page 166
- ◆ “NWConvertHandle” on page 168
- ◆ “NWCreateDirectory” on page 170
- ◆ “NWDeallocateDirectoryHandle” on page 173
- ◆ “NWDeleteDirectory” on page 175
- ◆ “NWDeleteTrustee” on page 177
- ◆ “NWDeleteTrusteeExt” on page 179
- ◆ “NWDeleteTrusteeFromDirectory” on page 181
- ◆ “NWFileServerFileCopy” on page 183

NWAddTrustee

Adds a trustee to the list of trustees in a file or directory

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT*, Windows* 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE)NWAddTrustee (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint32             objID,
    nuint16             rightsMask);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWAddTrustee
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pnstr8;
   objID : nuint32;
   rightsMask : nuint16
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare® server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired directory path (0 if the path parameter contains the complete path, including the volume name).

path

(IN) Points to the absolute path (or a path relative to the `dirHandle` parameter) of the directory to which a trustee is being added.

objID

(IN) Specifies the object ID for the object being added as a trustee.

rightsMask

(IN) Specifies the access rights mask being granted to the new trustee.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x8990	NO_FILES_AFFECTED_READ_ONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x8999	DIRECTORY_FULL
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FC	NO_SUCH_OBJECT
0x89FD	BAD_STATION_NUMBER
0x89FF	HARDWARE_FAILURE

Remarks

To modify a trustee rights list, the requesting workstation must have access control rights to the directory or to a parent of the directory.

If the object is already a trustee for the specified directory, the current access mask of the trustee is replaced by the value contained in the `rightsMask` parameter. Otherwise, the object is added as a trustee to the directory with rights equal to the `rightsMask` parameter.

NCP Calls

0x2222 23 17 Get File Server Information

0x2222 22 13 Add Trustee To Directory

0x2222 22 39 Add Extended Trustee To Directory Or File

0x2222 87 10 Add Trustee Set To File Or Subdirectory

See Also

[NWAddTrusteeToDirectory \(page 158\)](#), [NWScanNSDirectoryForTrustees \(page 530\)](#)

NWAddTrusteeExt

Adds a trustee to the list of trustees in a file or directory, using UTF-8 strings.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWAddTrusteeExt (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint32             objID,
    nuint16             rightsMask);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired directory path (0 if the `path` parameter contains the complete path, including the volume name).

path

(IN) Points to the absolute path (or a path relative to the `dirHandle` parameter) of the directory to which a trustee is being added. The characters in the string must be UTF-8.

objID

(IN) Specifies the object ID for the object being added as a trustee.

rightsMask

(IN) Specifies the access rights mask being granted to the new trustee. For possible values, see [“Trustee Rights” on page 124](#)

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x8990	NO_FILES_AFFECTED_READ_ONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x8999	DIRECTORY_FULL
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FC	NO_SUCH_OBJECT
0x89FD	BAD_STATION_NUMBER
0x89FF	HARDWARE_FAILURE

Remarks

To modify a trustee rights list, the requesting workstation must have access control rights to the directory or to a parent of the directory.

If the object is already a trustee for the specified directory, the current access mask of the trustee is replaced by the value contained in the `rightsMask` parameter. Otherwise, the object is added as a trustee to the directory with rights equal to the `rightsMask` parameter.

NCP Calls

0x2222 23 17 Get File Server Information
0x2222 22 13 Add Trustee To Directory
0x2222 22 39 Add Extended Trustee To Directory Or File
0x2222 87 10 Add Trustee Set To File Or Subdirectory
0x2222 89 10 Add Trustee Set To File Or Subdirectory

See Also

[NWAddTrustee \(page 153\)](#)

NWAddTrusteeToDirectory

Adds a trustee to the trustee list in a directory

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWAddTrusteeToDirectory (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint32             trusteeID,
    nuint8              rightsMask);
```

Delphi Syntax

```
uses calwin32;

Function NWAddTrusteeToDirectory
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pustr8;
   trusteeID : nuint32;
   rightsMask : nuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired directory path (0 if the path parameter contains the complete path, including the volume name).

path

(IN) Points to the absolute path (or a path relative to the directory handle) of the directory to which a trustee is being added.

trusteeID

(IN) Specifies the object ID for the object being added as a trustee.

rightsMask

(IN) Specifies the access rights mask the new trustee is being granted.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x8990	NO_FILES_AFFECTED_READ_ONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x8999	DIRECTORY_FULL
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FC	NO_SUCH_OBJECT
0x89FD	BAD_STATION_NUMBER
0x89FF	HARDWARE_FAILURE

Remarks

If the object is already a trustee for the specified directory, the current access mask of the trustee is replaced by the value contained in the `trusteeID` parameter. Otherwise, the object is added as a trustee to the directory and given a rights mask equal to the `trusteeID` parameter.

To modify a trustee rights list, the requesting workstation must have access control rights to the directory or to a parent of the directory.

The object must be static. If the object is dynamic, `NWAddTrusteeToDirectory` will return an error.

For Windows 32-bit platforms, `dirHandle` and `path` must be specified in the LONG namespace format.

For NLMs, `dirHandle` and `path` must be specified in the DOS namespace format, and `path` must be in upper case.

If you want to specify the name space that you are using for the parameters, use [NWAddTrusteeToNSDirectory \(page 448\)](#).

NCP Calls

0x2222 22 13 Add Trustee To Directory

0x2222 22 39 Trustee Add Ext

0x2222 23 17 Get File Server Information

0x2222 87 10 Add Trustee Set To File Or Subdirectory

See Also

[NWAddTrustee \(page 153\)](#), [NWAddTrusteeToNSDirectory \(page 448\)](#), [NWDeleteTrustee \(page 177\)](#), [NWDeleteTrusteeFromDirectory \(page 181\)](#), [NWDeleteTrusteeFromNSDirectory \(page 459\)](#), [NWScanNSDirectoryForTrustees \(page 530\)](#)

NWAllocPermanentDirectoryHandle

Allocates a permanent directory handle for a network directory

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY( NWCCODE )NWAllocPermanentDirectoryHandle (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *dirPath,
    NWDIR_HANDLE N_FAR *newDirHandle,
    puint8             effectiveRights);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWAllocPermanentDirectoryHandle
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const dirPath : pnstr8;
   Var newDirHandle : NWDIR_HANDLE;
   effectiveRights : puint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired directory path.

dirPath

(IN) Points to an absolute directory path (or a path relative to the `dirHandle` parameter) specifying the directory with which the new directory handle is to be associated (optional).

newDirHandle

(OUT) Points to the new directory handle.

effectiveRights

(OUT) Points to the effective rights of the directory trustee connected through the `dirHandle` parameter (optional).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x8999	DIRECTORY_FULL
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899D	NO_MORE_DIRECTORY_HANDLES
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	INVALID_DRIVE_NUMBER, HARDWARE_FAILURE

Remarks

To deallocate a permanent directory handle, call the `NWDeallocateDirectoryHandle` function.

If more than 255 handles are allocated, `NWAllocPermanentDirectoryHandle` may return a successful code; however, the `dirHandle` parameter will be zero.

NCP Calls

0x2222 22 03 Get Effective Directory Rights
0x2222 22 18 Alloc Permanent Directory Handle
0x2222 23 17 Get File Server Information
0x2222 87 12 Allocate Short Directory Handle

See Also

[NWAllocTempNSDirHandle2 \(page 451\)](#), [NWAllocTemporaryDirectoryHandle \(page 163\)](#), [NWDeallocateDirectoryHandle \(page 173\)](#)

NWAllocTemporaryDirectoryHandle

Assigns a temporary directory handle for the current name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWAllocTemporaryDirectoryHandle (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *dirPath,
    NWDIR_HANDLE N_FAR *newDirHandle,
    puint8             rightsMask);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWAllocTemporaryDirectoryHandle
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const dirPath : pnstr8;
   Var newDirHandle : NWDIR_HANDLE;
   rightsMask : puint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired directory path (or 0 if the dirPath parameter points to the complete path, including the volume name).

dirPath

(IN) Points to an absolute directory path (or a path relative to the NetWare directory handle) specifying the directory with which the new directory handle is associated.

newDirHandle

(OUT) Points to the new directory handle.

rightsMask

(OUT) Points to the effective rights of the directory trustee connected through the `newDirHandle` parameter (optional).

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x8999	DIRECTORY_FULL
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899D	NO_MORE_DIRECTORY_HANDLES
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	INVALID_DRIVE_NUMBER, HARDWARE_FAILURE

Remarks

The directory handles allocated by `NWAllocTemporaryDirectoryHandle` are automatically deallocated when the task terminates, or when the `NWDeallocateDirectoryHandle` function is called.

If more than 255 handles are allocated, `NWAllocTemporaryDirectoryHandle` may return a successful code; however, the `dirHandle` parameter will be zero.

NCP Calls

- 0x2222 22 03 Get Effective Directory Rights
- 0x2222 22 19 Allocate Temporary Directory Handle
- 0x2222 23 17 Get File Server Information
- 0x2222 87 12 Allocate Short Directory Handle

See Also

[NWAllocPermanentDirectoryHandle \(page 161\)](#), [NWAllocTempNSDirHandle2 \(page 451\)](#),
[NWDeallocateDirectoryHandle \(page 173\)](#)

NWConvertFileHandle

Converts a file handle to a 4- or 6-byte NetWare handle

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwmisc.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWConvertFileHandle (
    NWFIL_HANDLE fileHandle,
    nuint16      handleType,
    pnuint8      NWHandle,
    NWCONN_HANDLE N_FAR *conn);
```

Delphi Syntax

```
uses calwin32

Function NWConvertFileHandle
  (fileHandle : NWFIL_HANDLE;
   handleType : nuint16;
   NWHandle   : pnuint8;
   Var conn   : NWCONN_HANDLE
  ) : NWCCODE;
```

Parameters

fileHandle

(IN) Specifies the name of the local file handle to be converted to a NetWare handle.

handleType

(IN) Specifies the type of handle to create:

4 = Create a 4-byte NetWare handle

6 = Create a 6-byte NetWare handle

NWHandle

(OUT) Points to a 4- or 6-byte NetWare Handle to which the local file handle is being converted.

conn

(OUT) Points to the connection for which the NetWare handle is valid (optional).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x0006	INVALID_HANDLE
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8988	INVALID_FILE_HANDLE

Remarks

The handle returned by `NWConvertFileHandle` should not be used to call the `NWConvertHandle` function. Otherwise, a new OS file handle will be created.

If `NWConvertFileHandle` is called with only the NETX shell running, `INVALID_CONNECTION` will be returned. However, the NetWare handle will still be valid and the `conn` parameter will be set to zero.

If a pointer is passed in the `conn` parameter and the NETX shell is running, a valid NetWare handle will be returned as well as `0x8801`.

When a connection handle is obtained, a new licensed connection handle will be created. Close the new connection handle by calling the `NWCCCloseConn` function.

See Also

[NWConvertHandle \(page 168\)](#)

NWConvertHandle

Converts a NetWare handle to a local file handle

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwmisc.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWConvertHandle (
    NWCONN_HANDLE      conn,
    nuint8              accessMode,
    const void N_FAR   *NWHandle,
    nuint16             handleSize,
    nuint32             fileSize,
    NWFILE_HANDLE N_FAR *fileHandle);
```

Delphi Syntax

```
uses calwin32

Function NWConvertHandle
  (conn : NWCONN_HANDLE;
   accessMode : nuint8;
   const NWHandle : nptr;
   handleSize : nuint16;
   fileSize : nuint32;
   Var fileHandle : NWFILE_HANDLE
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the connection where the NetWare handle exists to which the local file handle is being converted.

accessMode

(IN) Specifies the type of access the user will have to the newly created file handle.

NWHandle

(IN) Points to the 4- or 6-byte NetWare handle being converted to a local file handle.

handleSize

(IN) Specifies the number of bytes in the NetWare handle; either 4 or 6.

fileSize

(IN) Specifies the number of bytes in the file being converted.

fileHandle

(OUT) Points to the local file handle created by NWConvertHandle.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
--------	------------

Remarks

The handle returned by the NWConvertFileHandle function should not be used to call NWConvertHandle. Otherwise, a new OS file handle will be created.

The file handle returned is appropriate for the platform for which the function is written. The file handle may be used for access to the attribute value including closing the file as well as reading and writing to the file.

See [Section 20.1, “Access Right Values,” on page 593](#) for the possible values for the `accessMode` parameter.

Call the file access functions that are native to your platform.

See Also

[NWConvertFileHandle \(page 166\)](#)

NWCreateDirectory

Creates a NetWare directory on the specified server

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWCreateDirectory (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *dirPath,
    nuint8              accessMask);
```

Delphi Syntax

```
uses calwin32;

Function NWCreateDirectory
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const dirPath : pnstr8;
   accessMask : nuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the root directory for the new directory (0 if the dirPath parameter points to the complete path, including the volume name).

dirPath

(IN) Points to the string containing the name and path of the new directory.

accessMask

(IN) Specifies the access rights mask for the new directory.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8984	NO_CREATE_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x8999	DIRECTORY_FULL
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899E	INVALID_FILENAME
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	HARDWARE_FAILURE (directory/file already exists)

Remarks

The `accessMask` parameter can be set using one or more of the following:

Hex	Definition
0xFB	TA_ALL
0x01	TA_READ
0x02	TA_WRITE
0x04	TA_OPEN
0x08	TA_CREATE
0x10	TA_DELETE
0x20	TA_OWNERSHIP
0x40	TA_SEARCH
0x80	TA_MODIFY

NOTE: Actual rights are set according to inherited rights.

NCP Calls

0x2222 22 10 Create Directory

0x2222 23 17 Get File Server Information

0x2222 87 01 Open Create File Or Subdirectory

See Also

[NWDeleteDirectory \(page 175\)](#)

NWDeallocateDirectoryHandle

Deallocates a directory handle allocated by `NWAllocTemporaryDirectoryHandle` or `NWAllocPermanentDirectoryHandle`

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWDeallocateDirectoryHandle (
    NWCONN_HANDLE    conn,
    NWDIR_HANDLE     dirHandle);
```

Delphi Syntax

```
uses calwin32

Function NWDeallocateDirectoryHandle
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle to be deallocated.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION

0x890A	NLM_INVALID_CONNECTION
0x899B	BAD_DIRECTORY_HANDLE

Remarks

When a workstation terminates or logs out, all directory handles for the workstation are deleted.

NCP Calls

0x2222 22 20 Deallocate Directory Handle

See Also

[NWAllocPermanentDirectoryHandle \(page 161\)](#), [NWAllocTempNSDirHandle2 \(page 451\)](#), [NWAllocTemporaryDirectoryHandle \(page 163\)](#), [NWGetDirectoryHandlePath \(page 191\)](#)

NWDeleteDirectory

Deletes a NetWare directory

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWDeleteDirectory (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *dirPath);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWDeleteDirectory
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const dirPath : pnstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the target directory root (0 if the `dirPath` parameter contains the complete path, including the volume name).

dirPath

(IN) Points to the string containing the path (relative to the `dirHandle` parameter) of the directory being deleted.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x898A	NO_DELETE_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899F	DIRECTORY_ACTIVE
0x89A0	DIRECTORY_NOT_EMPTY
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure

NCP Calls

0x2222 22 11 Delete Directory
0x2222 23 17 Get File Server Information
0x2222 87 08 Delete A File Or Subdirectory

See Also

[NWCreateDirectory](#) (page 170)

NWDeleteTrustee

Removes a trustee from the specified directory or a trustee list for a file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWDeleteTrustee (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *dirPath,
    nuint32             objID);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle for the directory whose trustee list is being deleted (0 if the `dirPath` parameter points to the complete path, including the volume name).

dirPath

(IN) Points to the directory from which the trustee is being removed.

objID

(IN) Specifies the object ID for the trustee being deleted.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION

0x898C	NO_MODIFY_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x8999	DIRECTORY_FULL
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FC	NO_SUCH_OBJECT
0x89FD	BAD_STATION_NUMBER
0x89FE	TRUSTEE_NOT_FOUND
0x89FF	HARDWARE_FAILURE, Failure

Remarks

NWDeleteTrustee also revokes the rights of the trustee in the specified directory.

To delete a trustee, the requesting workstation must have access control rights in the directory or in a parent directory.

Deleting the explicit assignment of an trustee object in a directory is not the same as assigning no rights to the object in the directory. If no rights are assigned in a directory, the object inherits the same rights as the parent directory.

NCP Calls

0x2222 22 14 Delete Trustee From Directory
0x2222 22 43 Trustee Remove Ext
0x2222 23 17 Get File Server Information
0x2222 87 11 Delete Trustee Set From File Or Subdirectory

See Also

[NWAddTrustee \(page 153\)](#), [NWDeleteTrusteeExt \(page 179\)](#), [NWIntScanForTrustees \(page 244\)](#), [NWScanNSDirectoryForTrustees \(page 530\)](#), [NWParseNetWarePath \(page 622\)](#)

NWDeleteTrusteeExt

Removes a trustee from the specified directory or a trustee list for a file, using UTF-8 strings.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWDeleteTrusteeExt (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *dirPath,
    nuint32             objID);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle for the directory whose trustee list is being deleted (0 if the `dirPath` parameter points to the complete path, including the volume name).

dirPath

(IN) Points to the absolute path (or a path relative to the `dirHandle` parameter) of the directory from which the trustee is being removed. The characters in the string must be UTF-8.

objID

(IN) Specifies the object ID for the trustee being deleted.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
--------	------------

0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x8999	DIRECTORY_FULL
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FC	NO_SUCH_OBJECT
0x89FD	BAD_STATION_NUMBER
0x89FE	TRUSTEE_NOT_FOUND
0x89FF	HARDWARE_FAILURE, Failure

Remarks

NWDeleteTrusteeExt also revokes the rights of the trustee in the specified directory.

To delete a trustee, the requesting workstation must have access control rights in the directory or in a parent directory.

Deleting the explicit assignment of an trustee object in a directory is not the same as assigning no rights to the object in the directory. If no rights are assigned in a directory, the object inherits the same rights as the parent directory.

NCP Calls

0x2222 22 14 Delete Trustee From Directory

0x2222 22 43 Trustee Remove Ext

0x2222 23 17 Get File Server Information

0x2222 87 11 Delete Trustee Set From File Or Subdirectory

0x2222 89 11 Delete Trustee Set From File Or Subdirectory

See Also

[NWDeleteTrustee \(page 177\)](#)

NWDeleteTrusteeFromDirectory

Removes a trustee from a directory trustee list

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWDeleteTrusteeFromDirectory (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint32             objID);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWDeleteTrusteeFromDirectory
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pustr8;
   objID : nuint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle for the directory whose trustee list is being modified (zero if the `path` parameter points to the complete path, including the volume name).

path

(IN) Points to an absolute path (or a path relative to the `dirHandle` parameter) specifying the directory from which the trustee is being removed.

objID

(IN) Specifies the object ID for the trustee being deleted.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
--------	------------

Remarks

NWDeleteTrusteeFromDirectory revokes the rights for a trustee in a specific directory. The requesting workstation must have access control rights in the directory or in a parent directory to delete a trustee.

Deleting the explicit assignment of an trustee object in a directory is not the same as assigning no rights to the object in the directory. If no rights are assigned in a directory, the object inherits the same rights it has in the parent directory.

If you want to specify the name space that you are using for the parameters, use [NWDeleteTrusteeFromNSDirectory \(page 459\)](#).

NCP Calls

0x2222 22 14 Delete Trustee From Directory

0x2222 22 43 Trustee Remove Ext

0x2222 23 17 Get File Server Information

0x2222 87 11 Delete Trustee Set From File Or Subdirectory

See Also

[NWAddTrusteeToDirectory \(page 158\)](#), [NWAddTrusteeToNSDirectory \(page 448\)](#), [NWDeleteTrusteeFromNSDirectory \(page 459\)](#), [NWParseNetWarePath \(page 622\)](#), [NWScanDirectoryForTrustees2 \(page 262\)](#), [NWScanNSDirectoryForTrustees \(page 530\)](#)

NWFileServerFileCopy

Copies a file or portion of a file from a source to a destination on the same NetWare server

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include<nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWFileServerFileCopy (
    NWFIL_HANDLE    srcFileHandle,
    NWFIL_HANDLE    dstFileHandle,
    nuInt32         srcOffset,
    nuInt32         dstOffset,
    nuInt32         bytesToCopy,
    pnuInt32        bytesCopied);
```

Delphi Syntax

```
uses calwin32

Function NWFileServerFileCopy
  (srcFileHandle : NWFIL_HANDLE;
   dstFileHandle : NWFIL_HANDLE;
   srcOffset : nuInt32;
   dstOffset : nuInt32;
   bytesToCopy : nuInt32;
   bytesCopied : pnuInt32
  ) : NWCCODE;
```

Parameters

srcFileHandle

(IN) Specifies the source file handle (index).

dstFileHandle

(IN) Specifies the destination file handle (index).

srcOffset

(IN) Specifies the offset in the source file where the copying is to begin.

dstOffset

(IN) Specifies the offset in the destination file where the copying is to begin.

bytesToCopy

(IN) Specifies the maximum number of bytes to copy.

bytesCopied

(OUT) Points to the number of bytes actually copied, or the size of a new destination file (optional).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x0006	Invalid File Handle
0x8830	NOT_SAME_CONNECTION
0x8901	ERR_INSUFFICIENT_SPACE
0x8983	IO_ERROR_NETWORK_DISK
0x8988	INVALID_FILE_HANDLE
0x8993	NO_READ_PRIVILEGES
0x8994	NO_WRITE_PRIVILEGES_OR_READONLY
0x8995	FILE_DETACHED
0x8996	SERVER_OUT_OF_MEMORY
0x89A2	READ_FILE_WITH_RECORD_LOCKED

Remarks

NWFileServerFileCopy is very efficient since the data does not come to the workstation; the server handles the duplication of the data internally.

If the source and destination files do not reside on the same server, NOT_SAME_CONNECTION is returned.

You must pass OS file handles in the `srcFileHandle` and `dstFileHandle` parameters. Use the appropriate OS functions that create and open files to return the file handles, depending on whether the destination file is a new or an existing file.

If the destination file is new, the `bytesCopied` parameter points to the size of the destination file. Otherwise, it points to the number of bytes copied.

To copy the entire source file, specify a value that matches or exceeds the file size in the `bytesToCopy` parameter.

NCP Calls

0x2222 74 Copy From One File To Another

10.3 NWGet* Functions

Click on any function name in the table of contents to view the purpose, syntax, parameters, and return values for that function.

- ◆ [“NWGetCompressedFileLengths” on page 186](#)
- ◆ [“NWGetDirectoryEntryNumber” on page 188](#)
- ◆ [“NWGetDirectoryHandlePath” on page 191](#)
- ◆ [“NWGetDirSpaceInfo” on page 193](#)
- ◆ [“NWGetDirSpaceLimitList” on page 195](#)
- ◆ [“NWGetDirSpaceLimitList2” on page 197](#)
- ◆ [“NWGetDiskIOsPending” on page 199](#)
- ◆ [“NWGetEffectiveRights” on page 200](#)
- ◆ [“NWGetEffectiveRightsExt” on page 203](#)
- ◆ [“NWGetExtendedFileAttributes2” on page 206](#)
- ◆ [“NWGetFileConnectionID” on page 209](#)
- ◆ [“NWGetFileDirEntryNumber” on page 211](#)
- ◆ [“NWGetSparseFileBitMap” on page 214](#)
- ◆ [“NWGetVolumeFlags” on page 216](#)

NWGetCompressedFileLengths

Returns information about the lengths of a compressed file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfinfo.h>

int NWGetCompressedFileLengths (
    int    handle,
    LONG   *uncompressedLength,
    LONG   *compressedLength;
```

Parameters

handle

(IN) Specifies the file handle for which to return the lengths.

uncompressedLength

(OUT) Points to the length of the file in an uncompressed state.

compressedLength

(OUT) Points to the length of the file after being compressed.

Return Values

0	Success
0xFF	Failure

Remarks

NWGetCompressedFileLengths returns information about the lengths of a compressed file.

If `handle` represents a file that is not compressed, the lengths will be invalid.

`uncompressedLength` specifies the length normally seen in directory listings.

The following code will open the file and enable it to be read without decompression:

```
#include <nwfileng.h>
#include <nwfattr.h>
```

```

#include <fcntl.h>
#include <sys/stat.h>
#include <nwinfo.h>void main()
{
    int handle;
    LONG uncom, com;

    handle=FEsopen("sys:\\compress\\test",O_RDONLY,H_DENYWR,S_IREAD,
        ENABLE_IO_ON_COMPRESSED_DATA_BIT,
PrimaryDataStream);
    NWGetCompressedFileLengths(handle, &uncom, &com);
    printf("The compressed size is %d and the uncompressed size is %d.",
com,
        uncom);
    close (handle);
}

```

The important parameter to FEsopen is S_IREAD, ENABLE_IO_ON_COMPRESSED_DATA_BIT. If this bit is not set, NWGetCompressedFileLengths uncompresses the file as it is read, which causes the resulting data to be inaccurate and leaves the file in an uncompressed state.

See Also

[NWSetCompressedFileLengths \(page 267\)](#)

NWGetDirectoryEntryNumber

Returns file information for a specified file under DOS and the name space associated with the specified directory handle

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY (NWCCODE) NWGetDirectoryEntryNumber (
    NWCONN_HANDLE    conn,
    nuint8           dirHandle,
    puint32          volumeNum,
    puint32          directoryEntry,
    puint32          DOSDirectoryEntry,
    puint32          nameSpace,
    puint32          parentDirEntry,
    puint32          parentDOSDirEntry);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetDirectoryEntryNumber
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   volumeNum : puint32;
   directoryEntry : puint32;
   DOSDirectoryEntry : puint32;
   nameSpace : puint32;
   parentDirEntry : puint32;
   parentDOSDirEntry : puint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the one byte directory handle.

volumeNum

(OUT) Points to the volume number of the directory handle.

directoryEntry

(OUT) Points to the directory entry number in the name space associated with the `dirHandle` parameter.

DOSDirectoryEntry

(OUT) Points to the directory entry number in the DOS name space.

nameSpace

(OUT) Points to the name space associated with the `directoryEntry` and `parentDirEntry` parameters.

parentDirEntry

(OUT) Points to the parent directory entry number of the directory handle in the name space associated with the `dirHandle` parameter.

parentDOSDirEntry

(OUT) Points to the parent directory entry number of the directory handle in the DOS name space.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION

Remarks

NWGetDirectoryEntryNumber returns the volume number, directory entry numbers, parent directory entry numbers in the DOS name space, and the name space associated with the directory handle.

One way to create the directory handle is to call the `NWAllocTempNSDirHandle2` function. If you specify a long directory name, the created directory handle will be associated with the LONG name space. If a DOS directory name is specified, the created directory handle will be associated with the DOS name space.

The `nameSpace` parameter can have the following values:

- 0 NW_NS_DOS
- 1 NW_NS_MAC
- 2 NW_NS_NFS
- 3 NW_NS_FTAM
- 4 NW_NS_LONG

NCP Calls

87 31 Get File Information

See Also

[NWAllocTempNSDirHandle2 \(page 451\)](#)

NWGetDirectoryHandlePath

Returns the path name of the directory associated with the given directory handle

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY( NWCCODE )NWGetDirectoryHandlePath (
    NWCONN_HANDLE   conn,
    NWDIR_HANDLE    dirHandle,
    pnstr8           dirPath);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetDirectoryHandlePath
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   dirPath : pnstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle for the directory whose path is to be reported.

dirPath

(OUT) Points to the directory path name associated with the `dirHandle` parameter.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8996	SERVER_OUT_OF_MEMORY
0x899B	BAD_DIRECTORY_HANDLE
0x89A1	DIRECTORY_IO_ERROR

Remarks

NWGetDirectoryHandlePath allows a client to retrieve the full directory path of the directory indexed by the `dirHandle` parameter. The string accessed by the `dirPath` parameter contains a path name in the following format:

```
Volume Name:Directory\Subdirectory\....
```

The string accessed by the `dirPath` parameter does not contain the name of the server. Its maximum length is 255 bytes.

Under NETX, if an invalid connection handle is passed to the `conn` parameter, NWGetDirectoryHandlePath will return 0x0000. An error will never be returned by NETX since NETX always chooses a default connection handle if the connection handle cannot be resolved.

NETX tries to resolve the connection ID through the preferred server first. If a preferred server does not exist, the request is directed to the default server (or the server implied by the default drive). If the default drive is mapped to a local drive, the shell directs the request to the primary server as the lowest connection priority.

NCP Calls

0x2222 22 01 Get Directory Path

See Also

[NWAllocTemporaryDirectoryHandle \(page 163\)](#), [NWDeallocateDirectoryHandle \(page 173\)](#)

NWGetDirSpaceInfo

Returns information on space usage for a volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY( NWCCODE )NWGetDirSpaceInfo (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    nuint16             volNum,
    DIR_SPACE_INFO     N_FAR *spaceInfo);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetDirSpaceInfo
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   volNum : nuint16;
   Var spaceInfo : DIR_SPACE_INFO
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle (nuint16).

dirHandle

(IN) Specifies the directory handle associated with the desired directory path (0 if volume information is to be returned).

volNum

(IN) Specifies the volume number to return space information for (0 if directory information is to be returned).

spaceInfo

(OUT) Points to the DIR_SPACE_INFO structure.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

If the `dirHandle` parameter is zero, `NWGetDirSpaceInfo` returns the volume information to the `DIR_SPACE_INFO` structure. Pass the volume number in `volNum`, which is obtained from calling `NWGetVolumeNumber`.

`purgeableBlocks` and `nonYetPurgeableBlocks` are set to 0 if the `dirHandle` parameter contains a nonzero value.

The `availableBlocks` field is the only field that returns information when disk space restrictions are in effect. The rest of the structure fields contain volume-wide information. If disk space restrictions are not in effect, the `availableBlocks` field will contain the number of blocks available for use on the entire volume.

One block equals the size of the block size for the specified volume, which is obtained by multiplying `sectorsPerBlock` by 512 bytes.

You can call [NWGetExtendedVolumeInfo](#) (*Volume Services*) to return the block size (in bytes).

NCP Calls

0x2222 22 44 Get Volume Purge Information

0x2222 22 45 Get Dir Info

See Also

[NWGetVolumeNumber](#) (Volume Management)

NWGetDirSpaceLimitList

Determines the actual space limitations for a directory

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY( NWCCODE )NWGetDirSpaceLimitList (
    NWCONN_HANDLE   conn,
    NWDIR_HANDLE    dirHandle,
    puint8          returnBuf);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetDirSpaceLimitList
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   returnBuf : puint8
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle pointing to the desired directory.

returnBuf

(OUT) Points to a 512-byte buffer containing the returned space list.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
--------	------------

Remarks

To find the actual amount of space available to a directory, scan all of the `current` fields and use the smallest one. You must omit values of `0x7FFFFFFF` and convert values that are larger than `0x7FFFFFFF` to zero. If no entries are returned, no space restrictions exist for the specified directory.

NOTE: All restrictions are returned in units of 4K blocks.

`returnBuf` points to a buffer holding the space limit information for the directory specified by `dirHandle`. This information is given in the order specified by the [NW_LIMIT_LIST \(page 349\)](#) structure.

IMPORTANT: `returnBuf` is not directly type compatible with the `NW_LIMIT_LIST` structure. It is highly recommended that instead of calling `NWGetDirSpaceLimitList`, applications now call [NWGetDirSpaceLimitList2 \(page 197\)](#), which uses a pointer to an `NW_LIMIT_LIST` structure.

See Also

[NWGetDirSpaceLimitList2 \(page 197\)](#), [NWSetDirSpaceLimit \(page 281\)](#)

NWGetDirSpaceLimitList2

Returns the actual space limitations for a directory.

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWGetDirSpaceLimitList2 (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    NW_LIMIT_LIST N_FAR *limitList);
```

Delphi Syntax

```
uses calwin32

Function NWGetDirSpaceLimitList2
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   Var limitList : NW_LIMIT_LIST
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle pointing to the desired directory.

limitList

(OUT) Points to NW_LIMIT_LIST.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
--------	------------

Remarks

To find the actual amount of space available to a directory, scan all of the `current` fields and use the smallest one. You must omit values of `0x7FFFFFFF` and convert values that are larger than `0x7FFFFFFF` to zero. If no entries are returned, no space restrictions exist for the specified directory.

All restrictions are returned in units of 4K blocks.

NOTE: If you use this function in a loop on an NSS volume, server utilization can rise to 100% which causes a denial of service to connections. You need to limit the number of quick calls to this function to under 200 and then let the server utilization drop before calling another set.

Server utilization is not affected by numerous quick calls to this function on traditional volumes.

NCP Calls

0x2222 22 35 Get Directory Disk Space Restriction

See Also

[NWSetDirSpaceLimit \(page 281\)](#)

NWGetDiskIOsPending

Returns the number of pending disk IOs the server has at the specified point in time

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwinfo.h>

int NWGetDiskIOsPending (
    void);
```

Return Values

Returns the number of pending disk IOs the server has upon successful completion.

Remarks

The value returned by NWGetDiskIOsPending is the same as the value for "Current disk requests" as reported by the MONITOR.NLM file.

NWGetEffectiveRights

Returns effective rights for the specified directory

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWGetEffectiveRights (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    pnuint16           effectiveRights);
```

Delphi Syntax

```
uses calwin32;

Function NWGetEffectiveRights
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pustr8;
   effectiveRights : pnuint16
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle associated with the directory path for which the effective rights are desired (0 if the `path` parameter points to the complete path, including the volume name).

path

(IN) Points to the absolute path (or a path relative to the `dirHandle` parameter) of the directory whose effective rights mask is being returned.

effectiveRights

(OUT) Points to the effective rights mask for the directory.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure

Remarks

To determine the effective rights of the requesting workstation, `NWGetEffectiveRights` performs a logical AND between the maximum rights mask of the directory and the current trustee rights of the workstation.

The current trustee rights are obtained by performing a logical OR between a trustee access mask and the trustee access mask of any object to which the process is security equivalent.

The current trustee rights can be explicitly listed in the directory or inherited from the parent directory. The maximum rights masks of parent directories do not affect inherited trustee rights.

The `effectiveRights` parameter returned to the client indicates which of the eight possible directory rights the client has in the targeted directory. An `effectiveRights` parameter of zero indicates the client has no rights in the target directory.

The maximum rights mask bits are defined in the table below:

C Value	Delphi Value	Value Description
0x0001	\$0001	TR_READ
0x0002	\$0002	TR_WRITE
0x0008	\$0008	TR_CREATE
0x0010	\$0010	TR_DELETE
0x0010	\$0020	TR_OWNERSHIP
0x0040	\$0040	TR_FILE_SCAN

C Value	Delphi Value	Value Description
0x0080	\$0080	TR_MODIFY

NWGetEffectiveRights works on files as well as directories.

See [effright.c \(../../../../samplecode/clib_sample/file/effright/effright.c.html\)](#) for sample code.

NCP Calls

0x2222 22 3 Get Effective Directory Rights

0x2222 22 42 Get Effective Rights

0x2222 23 17 Get File Server Information

0x2222 87 29 Get Effective Directory Rights

NWGetEffectiveRightsExt

Returns effective rights for the specified directory

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWGetEffectiveRightsExt (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    pnuint16           effectiveRights);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle associated with the directory path for which the effective rights are desired (0 if the `path` parameter points to the complete path, including the volume name).

path

(IN) Points to the absolute path (or a path relative to the `dirHandle` parameter) of the directory whose effective rights mask is being returned. The characters in the string must be UTF-8.

effectiveRights

(OUT) Points to the effective rights mask for the directory. (See Remarks for a list of values.)

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure

Remarks

To determine the effective rights of the requesting workstation, `NWGetEffectiveRightsExt` performs a logical AND between the maximum rights mask of the directory and the current trustee rights of the workstation.

The current trustee rights are obtained by performing a logical OR between a trustee access mask and the trustee access mask of any object to which the process is security equivalent.

The current trustee rights can be explicitly listed in the directory or inherited from the parent directory. The maximum rights masks of parent directories do not affect inherited trustee rights.

The `effectiveRights` parameter returned to the client indicates which of the eight possible directory rights the client has in the targeted directory. An `effectiveRights` parameter of zero indicates the client has no rights in the target directory.

The maximum rights mask bits are defined in the table below:

C Value	Value Description
0x0001	TR_READ
0x0002	TR_WRITE
0x0008	TR_CREATE
0x0010	TR_DELETE
0x0010	TR_OWNERSHIP
0x0040	TR_FILE_SCAN
0x0080	TR_MODIFY

`NWGetEffectiveRightsExt` works on files as well as directories.

NCP Calls

0x2222 22 3 Get Effective Directory Rights

0x2222 22 42 Get Effective Rights

0x2222 23 17 Get File Server Information

0x2222 87 29 Get Effective Directory Rights

0x2222 89 29 Get Effective Directory Rights

NWGetExtendedFileAttributes2

Returns the NetWare extended file attributes for the specified file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include<nwfile.h>
or
#include<nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWGetExtendedFileAttributes2 (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    puint8              extAttrs);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetExtendedFileAttributes2
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pustr8;
   extAttrs : puint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the new root directory.

path

(IN) Points to the string containing the name and path of the new directory.

extAttrs

(OUT) Points to the extended attributes of the file.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8988	INVALID_FILE_HANDLE
0x8989	NO_SEARCH_PRIVILEGES
0x8993	NO_READ_PRIVILEGES
0x8994	NO_WRITE_PRIVILEGES_OR_READONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	NO_FILES_FOUND_ERROR

Remarks

NWGetExtendedFileAttributes2 requires Search rights to the directory where the file resides.

The `path` parameter can specify the complete path name or a path relative to the current working directory. For example, if the complete path name is `SYS:ACCOUNT/DOMEST/TARGET.DAT` and the directory handle mapping is `SYS:ACCOUNT`, the `path` parameter could be the following:

```
SYS:ACCOUNT/DOMEST/TARGET.DAT or  
DOMEST/TARGET.DAT
```

The information accessed by the `extAttrs` parameter is interpreted as follows:

0-2	Search mode bits
4	Transaction bit
5	Index bit
6	Read audit bit
7	Write audit bit

NCP Calls

0x2222 23 15 Scan File Information

See Also

[NWSetExtendedFileAttributes2 \(page 283\)](#)

NWGetFileConnectionID

Returns the connection handle of the server owning the specified file handle

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWGetFileConnectionID (
    NWFHANDLE_HANDLE fileHandle,
    NWCONN_HANDLE N_FAR *conn);
```

Delphi Syntax

```
uses calwin32

Function NWGetFileConnectionID
  (fileHandle : NWFHANDLE;
   Var conn : NWCONN_HANDLE
  ) : NWCCODE;
```

Parameters

fileHandle

(IN) Specifies the file handle.

conn

(OUT) Points to the connection handle.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x89FD	UNKNOWN_REQUEST

Remarks

The server connection handle identifies a specific NetWare server to workstation connection.

NWGetFileConnectionID only works with VLMs loaded; it will not work with NETX. If NETX is loaded, UNKNOWN_REQUEST will be returned.

NWGetFileDirEntryNumber

Returns file information for a specified file under DOS and the name space associated with the specified file handle

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY (NWCCODE) NWGetFileDirEntryNumber (
    NWFIL_HANDLE fileHandle,
    puint32      volumeNum,
    puint32      directoryEntry,
    puint32      DOSDirectoryEntry,
    puint32      nameSpace,
    puint32      dataStream,
    puint32      parentDirEntry,
    puint32      parentDOSDirEntry);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetFileDirEntryNumber
(fileHandle : NWFIL_HANDLE;
volumeNum : puint32;
directoryEntry : puint32;
DOSDirectoryEntry : puint32;
nameSpace : puint32;
dataStream : puint32;
parentDirEntry : puint32;
parentDOSDirEntry : puint32
) : NWCCODE;
```

Parameters

fileHandle

(IN) Specifies the file handle.

volumeNum

(OUT) Points to the volume number of the file handle.

directoryEntry

(OUT) Points to the directory entry number in the name space associated with the `fileHandle` parameter.

DOSDirectoryEntry

(OUT) Points to the directory entry number in the DOS name space.

nameSpace

(OUT) Points to the name space associated with the `directoryEntry` and `parentDirEntry` parameters.

dataStream

(OUT) Points to the data stream number if the name space is `NW_NS_MAC`:

1 Data fork

0 Resource fork and anything else

parentDirEntry

(OUT) Points to the parent directory entry number of the file handle in the name space associated with the `fileHandle` parameter.

parentDOSDirEntry

(OUT) Points to the parent directory entry number of the file handle in the DOS name space.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x0006	INVALID_HANDLE
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8988	INVALID_FILE_HANDLE

Remarks

`NWGetFileDirEntryNumber` returns the volume number, directory entry numbers, parent directory entry numbers in the DOS name space, and the name space associated with the file handle.

One way to create the file handle is to call the `NWOpenNSEntry` function. If you specify a long file name, the created file handle will be associated with the `LONG` name space. If a DOS file name is specified, the created file handle will be associated with the `DOS` name space.

The `nameSpace` parameter can have the following values:

0 `NW_NS_DOS`

1 `NW_NS_MAC`

2 `NW_NS_NFS`

3 NW_NS_FTAM
4 NW_NS_LONG

NCP Calls

87 31 Get File Information

See Also

[NWOpenNSEntry \(page 516\)](#)

NWGetSparseFileBitMap

Returns a bit map showing which blocks in a sparse file contain data

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY (NWCCODE) NWGetSparseFileBitMap (
    NWCONN_HANDLE    conn,
    nuint32           fileHandle,
    nint16            flag,
    nuint32           offset,
    pnuint32          blockSize,
    pnuint8           bitMap);
```

Delphi Syntax

```
uses calwin32

Function NWGetSparseFileBitMap
  (conn : NWCONN_HANDLE;
   fileHandle : NWFILE_HANDLE;
   flag : nint16;
   offset : nuint32;
   blockSize : pnuint32;
   bitMap : pnuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

fileHandle

(IN) Specifies the 4-byte OS or NetWare file handle. If a NetWare file handle is used, a connection handle must be passed.

flag

(IN) Specifies whether the `fileHandle` parameter contains an OS or NetWare handle.

offset

(IN) Specifies the starting offset of the bit map in bytes.

blockSize

(OUT) Points to the size of the allocation block.

bitMap

(OUT) Points to a 512-byte array to receive the bit map (1 bit for each block).

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8988	INVALID_FILE_HANDLE

Remarks

NWGetSparseFileBitMap contains one bit for each block in the sparse file. A one indicates there is data in the block; a zero indicates there isn't any data in the block.

Use the `conn` parameter when NETX is running or the `fileHandle` parameter contains a NetWare handle (otherwise ignored).

If the `flag` parameter is 0, the `fileHandle` parameter contains a 4-byte OS file handle. If the `flag` parameter is nonzero, the `fileHandle` parameter contains a 6-byte NetWare handle.

The `bitMap` parameter must point to an array of 512 bytes.

NCP Calls

0x2222 85 Get Sparse File Data Block Bit Map

NWGetVolumeFlags

Returns the flags currently set on the specified volume

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfileio.h>

LONG NWGetVolumeFlags (
    LONG volume,
    LONG *flags);
```

Parameters

volume

(IN) Specifies the volume to return attributes for.

flags

(OUT) Points to a the flags set for the specified volume.

Return Values

If NWGetVolumeFlags is successful, a pointer to the set flags is returned. Otherwise, -1 is returned.

Remarks

flags can have the following values:

0x02	SUB_ALLOCATION_FLAG: If set, sub allocation units are valid on this volume.
0x04	FILE_COMPRESSION_FLAGS: If set, file compression is enabled on this volume.
0x08	DATA_MIGRATION_FLAG: If set, data migration is allowed on this volume.
0x40	VOLUME_IMMEDIATE_PURGE_FLAG: If set, this volume's deleted files will be purged immediately.

See Also

[NWSetVolumeFlags \(page 292\)](#)

10.4 NWI*-NWR* Functions

Click on any function name in the table of contents to view the purpose, syntax, parameters, and return values for that function.

- ◆ [“NWIntEraseFiles” on page 218](#)
- ◆ [“NWIntFileSearchContinue” on page 221](#)
- ◆ [“NWIntFileSearchInitialize” on page 224](#)
- ◆ [“NWIntMoveDirEntry” on page 226](#)
- ◆ [“NWIntScanDirectoryInformation2” on page 229](#)
- ◆ [“NWIntScanDirEntryInfo” on page 232](#)
- ◆ [“NWIntScanExtendedInfo” on page 235](#)
- ◆ [“NWIntScanFileInformation2” on page 238](#)
- ◆ [“NWIntScanFileInformation2Ext” on page 241](#)
- ◆ [“NWIntScanForTrustees” on page 244](#)
- ◆ [“NWIntScanForTrusteesExt” on page 248](#)
- ◆ [“NWModifyMaximumRightsMask” on page 251](#)
- ◆ [“NWRenameDirectory” on page 254](#)
- ◆ [“NWRenameFile” on page 256](#)

NWIntEraseFiles

Deletes NetWare files from the server

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWIntEraseFiles (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              searchAttrs,
    nuint16             augmentFlag);
```

Delphi Syntax

```
uses calwin32

Function NWIntEraseFiles
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pustr8;
   searchAttrs : nuint8;
   augmentFlag : nuint16
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle containing the file to erase.

dirHandle

(IN) Specifies the directory handle of the file to be erased (0 if the `path` parameter contains the complete path including the volume name).

path

(IN) Points to the string containing the file path (including the file name) of the file to be erased.

searchAttrs

(IN) Specifies the search attributes.

augmentFlag

(IN) Specifies if wildcards are augmented:

0 = wildcards are not augmented

nonzero = wildcards are augmented

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x898A	NO_DELETE_PRIVILEGES
0x898D	SOME_FILES_AFFECTED_IN_USE
0x898E	NO_FILES_AFFECTED_IN_USE
0x898F	SOME_FILES_AFFECTED_READ_ONLY
0x8990	NO_FILES_AFFECTED_READ_ONLY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89FF	NO_FILES_FOUND_ERROR

Remarks

The `searchAttrs` parameter includes system and/or hidden files. If only the system bit is set in the `searchAttrs` parameter, all files are affected except hidden files. If only the hidden bit is set, all files are affected except system files. When neither bit is set (0x00), only files that are not designated either hidden or system are affected.

NOTE: A file is designated hidden or system if its corresponding file attribute is set.

Search attributes to use in finding a file follow:

0x00	none
0x02	FA_HIDDEN
0x04	FA_SYSTEM
0x06	both

The `path` parameter can specify either a complete path name or a path relative to the current working directory. For example, if the complete path name is `SYS:ACCOUNT/DOMEST/TARGET.DAT` and the directory handle mapping is `SYS:ACCOUNT`, the value of the `path` parameter could be either of the following:

`SYS:ACCOUNT/DOMEST/TARGET.DAT` or `DOMEST/TARGET.DAT`

The `path` parameter can point to wildcards in the file name only. Wildcard matching uses the method defined by the application when it passes a wildcard character.

The client must have file deletion privileges in the target directory or NWIntEraseFiles will fail.
If a file has the immediate purge attribute set, the file cannot be recovered.

NCP Calls

0x2222 23 17 Get File Server Information

0x2222 68 Erase File

0x2222 87 08 Delete A File Or Subdirectory

See Also

[NWPurgeDeletedFile \(page 46\)](#), [NWRecoverDeletedFile \(page 49\)](#), [NWRenameFile \(page 256\)](#)

NWIntFileSearchContinue

Iteratively retrieves all directory entries matching the `searchPath` parameter in the DOS name space

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWIntFileSearchContinue (
    NWCONN_HANDLE      conn,
    nuInt8              volNum,
    nuInt16             dirID,
    nuInt16             searchContext,
    nuInt8              searchAttr,
    const nstr8 N_FAR  *searchPath,
    pnuInt8             retBuf,
    nuInt16             augmentFlag);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWIntFileSearchContinue
  (conn : NWCONN_HANDLE;
   volNum : nuInt8;
   dirID : nuInt16;
   searchContext : nuInt16;
   searchAttr : nuInt8;
   const searchPath : pnstr8;
   retBuf : pnuInt8;
   augmentFlag : nuInt16
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNum

(IN) Specifies the volume number returned by the initialize function.

dirID

(IN) Specifies the directory ID returned by the initialize function.

searchContext

(IN) Specifies the sequence number returned by the NWIntFileSearchInitialize function.

searchAttr

(IN) Specifies the attributes to apply to the search.

searchPath

(IN) Points to the path (file name, directory name, or wildcard).

retBuf

(OUT) Points to the information returned by the server.

augmentFlag

(IN) Specifies if wildcards are augmented:

0 = wildcards are not augmented

nonzero = wildcards are augmented

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x89FF	NO_FILES_FOUND_ERROR

Remarks

NWIntFileSearchContinue returns two different search structures depending on whether the match is a directory or a file. The application is responsible for determining the type of match, or for limiting the search to files or directories only. The two search structures are [SEARCH_FILE_INFO](#) (page 366) and [SEARCH_DIR_INFO](#) (page 363).

On the first iteration, use the sequence number returned by the NWIntFileSearchInitialize function. For subsequent iterations, use the `sequenceNumber` field from the `SEARCH_FILE_INFO` or `SEARCH_DIR_INFO` structure.

Valid search attributes follow:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM

C Value	Delphi Value	Value Name
0x10	\$10	FA_DIRECTORY

If other values are used for search attributes, each will be treated as FA_NORMAL.

NCP Calls

0x2222 63 File Search Continue

NWIntFileSearchInitialize

Searches for files on a server

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWIntFileSearchInitialize (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    puint8              volNum,
    puint16             dirID,
    puint16             iterHnd,
    puint8              accessRights,
    nuint16             augmentFlag);
```

Delphi Syntax

```
uses calwin32

Function NWIntFileSearchInitialize
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   path : pnstr8;
   volNum : puint8;
   dirID : puint16;
   iterhandle : puint16;
   accessRights : puint8;
   augmentFlag : nuint16
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the base directory handle to search.

path

(IN) Points to the path (relative to `dirHandle`) on which to initialize the search.

volNum

(OUT) Points to the corresponding volume number.

dirID

(OUT) Points to the directory ID corresponding to the specified path.

iterHnd

(OUT) Points to a sequence number to be used in calling `NWIntFileSearchContinue` (initially -1).

accessRights

(OUT) Points to the access rights of the workstation to the specified directory.

augmentFlag

(IN) Is reserved (pass in zero).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

A value of 0 should be passed to the `dirHandle` parameter if the directory handle is not known. In the absence of the directory handle, the `path` parameter needs to specify the volume as well.

NCP Calls

0x2222 62 File Search Initialize

See Also

[NWIntFileSearchContinue \(page 221\)](#)

NWIntMoveDirEntry

Moves or renames a directory entry (file or directory) on the same server (same volume)

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWIntMoveDirEntry (
    NWCONN_HANDLE      conn,
    nuint8              searchAttrs,
    NWDIR_HANDLE        srcDirHandle,
    const nstr8 N_FAR  *srcPath,
    NWDIR_HANDLE        dstDirHandle,
    const nstr8 N_FAR  *dstPath,
    nuint16             augmentFlag);
```

Delphi Syntax

```
uses calwin32

Function NWIntMoveDirEntry
  (conn : NWCONN_HANDLE;
   searchAttrs : nuint8;
   srcDirHandle : NWDIR_HANDLE;
   const srcPath : pustr8;
   dstDirHandle : NWDIR_HANDLE;
   const dstPath : pustr8;
   augmentFlag : nuint16
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

searchAttrs

(IN) Specifies the attributes to use in searching for the source entries.

srcDirHandle

(IN) Specifies the directory handle for the source directory (not optional, cannot be zero).

srcPath

(IN) Points to the source path (wildcards are allowed).

dstDirHandle

(IN) Specifies the NetWare directory handle for the destination directory.

dstPath

(IN) Points to the path name to use for the destination entry.

augmentFlag

(IN) Specifies if wildcards are augmented:

0 = wildcards are not augmented

nonzero = wildcards are augmented

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8987	WILD_CARDS_IN_CREATE_FILE_NAME
0x898B	NO_RENAME_PRIVILEGES
0x898D	SOME_FILES_AFFECTED_IN_USE
0x898E	NO_FILES_AFFECTED_IN_USE "All files in use"
0x898F	SOME_FILES_AFFECTED_READ_ONLY
0x8990	NO_FILES_AFFECTED_READ_ONLY "Read-only access to volume"
0x8991	SOME_FILES_RENAMED_NAME_EXISTS
0x8992	NO_FILES_RENAMED_NAME_EXISTS
0x899A	RENAMING_ACROSS_VOLUMES
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A4	ERR_RENAME_DIR_INVALID
0x89FF	NO_FILES_FOUND_ERROR

Remarks

To call `NWIntMoveDirEntry`, you must have file modification privileges in both the source and the target directories.

The specified paths are relative to the specified directory handles. NetWare 3.11 and above accepts paths relative to the directory handle, as well as full paths that include the volume. If full names are

used, be careful that the maximum request length is not exceeded. Path names larger than 255 are not supported.

The `searchAttrs` parameter specifies the kind of entry to look for (hidden, system, etc.). If only the system bit is set, all files are affected except hidden files. If only the hidden bit is set, all files are affected except system files. When neither bit is set (0x00), only files that are not designated either hidden or system are affected.

The `searchAttrs` parameter can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x01	\$01	FA_READ_ONLY
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x08	\$08	FA_EXECUTE_ONLY
0x10	\$10	FA_DIRECTORY
0x20	\$20	FA_NEEDS_ARCHIVED
0x80	\$80	FA_SHAREABLE

A file is designated hidden or system if its corresponding file attribute is set.

The advantage of calling `NWIntMoveDirEntry` is its speed and efficiency. Since the move is within the server, the entry in the file system is simply deleted from the source and inserted in the destination. Moving directory entries occurs only on the file system level. There is no physical transfer of data between the source and the destination.

`NWIntMoveDirEntry` will move files within the same volume only. If you attempt to move a file across different volumes, `RENAMING_ACROSS_VOLUMES` is returned.

NOTE: If the mac namespace has been enabled on the volume, do not use `NWIntMoveDirEntry` to move files or directories.

NCP Calls

0x2222 23 17 Get File Server Information

0x2222 69 Rename File

0x2222 87 04 Rename Or Move A File Or Subdirectory

NWIntScanDirectoryInformation2

Returns directory information for a directory specified by the connection handle, directory handle, and directory path

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWIntScanDirectoryInformation2 (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *srchPath,
    puint8              sequence,
    pustr8              dirName,
    puint32             dirDateTime,
    puint32             ownerID,
    puint8              rightsMask,
    nuint16             augmentFlag);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWIntScanDirectoryInformation2
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   searchPath : pustr8;
   sequence : puint8;
   dirName : pustr8;
   dirDateTime : puint32;
   ownerID : puint32;
   rightsMask : puint8;
   augmentFlag : nuint16
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle for the directory being scanned.

srchPath

(IN) Points to an absolute directory path with a maximum length of 255 (or a path relative to the directory handle) and a search pattern (optional).

sequence

(IN/OUT) Points to a 9-byte sequence number to be used for subsequent calls (the first 4 bytes should be 0xFF initially).

dirName

(OUT) Points to the directory name found (256 bytes, optional).

dirDateTime

(OUT) Points to the creation date and time of the directory (4 bytes, optional) in the DOS date and time format.

ownerID

(OUT) Points to the object ID of the owner for the directory (optional).

rightsMask

(OUT) Points to the maximum rights mask for the directory found (optional).

augmentFlag

(IN) Specifies if wildcards are augmented:

0 = wildcards are not augmented

nonzero = wildcards are augmented

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89FF	NO_FILES_FOUND_ERROR

Remarks

All parameter fields must be filled. However, NULL may be substituted in parameters where no information is desired.

The `dirHandle` parameter can be zero if the `srchPath` parameter points to the complete path, including the volume name.

The string accessed by the `srchPath` parameter can include wildcard characters. If wildcards are used, only the directory information for the first matching directory is returned.

The `rightsMask` parameter can have the following values:

0x00 = TA_NONE
0x01 = TA_READ
0x02 = TA_WRITE
0x04 = TA_OPEN
0x08 = TA_CREATE
0x10 = TA_DELETE
0x20 = TA_OWNERSHIP
0x40 = TA_SEARCH
0x80 = TA_MODIFY
0xFB = TA_ALL

NOTE: TA_OPEN is obsolete in NetWare 3.x and above.

NCP Calls

0x2222 22 01 Get Directory Path
0x2222 22 02 Scan Directory Information
0x2222 23 17 Get File Server Information
0x2222 87 02 Initialize Search
0x2222 87 03 Search For File Or Subdirectory
0x2222 87 06 Obtain File Or Subdirectory Information

See Also

[NWParseNetWarePath \(page 622\)](#)

NWIntScanDirEntryInfo

Obtains information about NetWare 3.x, 4.x, 5.x, and 6.x directory entries (files or directories) in the DOS name space

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWIntScanDirEntryInfo (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    nuint16             attrs,
    puint32             iterHandle,
    const nuint8 N_FAR *searchPattern,
    NWENTRY_INFO N_FAR *entryInfo,
    nuint16             augmentFlag);
```

Delphi Syntax

```
uses calwin32

Function NWIntScanDirEntryInfo
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   attrs : nuint16;
   iterHandle : puint32;
   searchPattern : puint8;
   Var entryInfo : NWENTRY_INFO;
   augmentFlag : nuint16
  ) : NWCCODE ;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare DOS directory handle indexing the directory to scan (not optional, cannot be 0).

attrs

(IN) Specifies the attributes to be used for the scan.

iterHandle

(IN/OUT) Points to a uint32 buffer to receive the search sequence from the server.

searchPattern

(IN) Points to the name of the entry for which to scan (wildcards are allowed).

entryInfo

(OUT) Points to the NWINTRY_INFO structure (zeroed out initially).

augmentFlag

(IN) Specifies if wildcards are augmented:

0 = wildcards are not augmented

nonzero = wildcards are augmented

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8989	NO_SEARCH_PRIVILEGES
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89FF	NO_FILES_FOUND_ERROR

Remarks

NWIntScanDirEntryInfo can only be called with non-augmented wildcards if the `augmentFlag` parameter is set to 0. For example, `*.*` will match anything with a period, while `*` will match any string.

NWIntScanDirEntryInfo will support augmented wildcard characters if the `augmentFlag` parameter is set to 1 or if the high-order bits have been manually set. For example, `*` will now match zero or more characters up to a period or an end-of-string.

On the first call, the `iterHandle` parameter should point to 0xFFFFFFFF. After that, the server manages the information. All scanning is complete when the server returns 0x89FF.

The `searchPattern` parameter cannot point to any path elements and the `dirHandle` parameter must index the complete path.

NWIntScanDirEntryInfo can also be used to scan for information about other directories, including the root directory. In this mode, the `dirHandle` parameter needs to index the root or a directory, and the `searchPattern` parameter needs to point to NULL.

NWIntScanDirEntryInfo works with the DOS name space only. Path and file names must be upper cased. To scan using alternate name spaces, convert the path to a DOS name space by calling either the `NWGetNSPath` or `NWScanNSEntryInfo` function. You can also scan the Macintosh name space by calling the `NWAFPScanFileInformation` function.

The `attrs` parameter can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x10	\$10	FA_DIRECTORY

The `NWENTRY_INFO` structure should be initialized to 0 before `NWIntScanDirEntryInfo` is called for the first time.

NCP Calls

0x2222 22 01 Get Directory Path

0x2222 22 30 Scan A Directory

0x2222 22 31 Get Directory Entry

See Also

[NWAFPScanFileInformation](#) (Single and Intra-File Management), [NWGetNSInfo](#) (page 481), [NWIntScanExtendedInfo](#) (page 235), [NWScanNSEntryInfo](#) (page 533)

NWIntScanExtendedInfo

Scans a directory for the extended file information

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWIntScanExtendedInfo (
    NWCONN_HANDLE          conn,
    NWDIR_HANDLE           dirHandle,
    nuint8                  attrs,
    pnuint32                iterHandle,
    const nstr8 N_FAR       *searchPattern,
    NW_EXT_FILE_INFO N_FAR *entryInfo,
    nuint16                 augmentFlag);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWIntScanExtendedInfo
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   attrs : nuint8;
   iterHandle : pnuint32;
   const searchPattern : pnstr8;
   Var entryInfo : NW_EXT_FILE_INFO;
   augmentFlag : nuint16
  ) : NWCCODE ;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle for the directory to be scanned.

attrs

(IN) Specifies the search attributes.

iterHandle

(IN/OUT) Points to the search sequence number (-1 initially).

searchPattern

(IN) Points to the pattern for which to search (no wildcards are allowed).

entryInfo

(OUT) Points to the NW_EXT_FILE_INFO structure containing the extended file information.

augmentFlag

(IN) Specifies if wildcards are augmented:

0 = wildcards are not augmented

nonzero = wildcards are augmented

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8989	NO_SEARCH_PRIVILEGES
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89FF	NO_FILES_FOUND_ERROR

Remarks

NWIntScanExtendedInfo works only on files, not on directories.

All scanning is complete when the server returns 0x89FF.

NWIntScanExtendedInfo is synonymous with the NWIntScanDirEntryInfo function and uses an extension of the information structure.

The `iterHandle` parameter should point to 0xFFFFFFFF for the first call.

The `attrs` parameter is used to include system and/or hidden files. If only the system bit is set in the `attrs` parameter, all files are affected except hidden files. If only the hidden bit is set, all files are affected except system files. When neither bit is set (0x00), only files designated either hidden or system are affected.

NOTE: A file is designated hidden or system if its corresponding file attribute is set.

The `attrs` parameter can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x10	\$10	FA_DIRECTORY

The extended file information contains the information returned by the `NWIntScanDirEntryInfo` function plus the sizes of the data and resource forks. `NWIntScanExtendedInfo` also returns the physical size of a file.

NOTE: In the case of sparse files, the logical size may be much larger than the physical size.

NCP Calls

0x2222 22 40 Scan Directory Disk Space

See Also

[NWIntScanDirEntryInfo \(page 232\)](#), [NWScanNSEntryInfo \(page 533\)](#)

NWIntScanFileInformation2

Scans the specified directory for the specified file (or directory) and returns the associated directory entry information in the DOS name space

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWIntScanFileInformation2 (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *filePattern,
    nuint8             searchAttrs,
    pnuint8            iterHandle,
    NW_FILE_INFO2 N_FAR *info,
    nuint16            augmentFlag);
```

Delphi Syntax

```
uses calwin32

Function NWIntScanFileInformation2
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const filePattern : pnstr8;
   searchAttrs : nuint8;
   iterHandle : pnuint8;
   Var info : NW_FILE_INFO2;
   augmentFlag : nuint16;
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle relative to the `filePattern` parameter (or 0 if the `filePattern` parameter points to the complete path, including the volume name).

filePattern

(IN) Points to the string containing the file name or wildcard pattern to use in the search.

searchAttrs

(IN) Specifies the attributes to use for searching.

iterHandle

(IN/OUT) Inputs a pointer to the sequence number (set the first 4 bytes to 0xFF initially).
Outputs a pointer to the 9-byte sequence number to be used for subsequent iterations.

info

(OUT) Points to the NW_FILE_INFO2 structure containing the file information.

augmentFlag

(IN) Specifies if wildcards are augmented:

0 = wildcards are not augmented

nonzero = wildcards are augmented

Note that if the high-order bit of a wildcard character is 1, NetWare interprets that character as being a DOS wildcard (which is also called an augmented wildcard) and uses DOS rules for interpretation of that wildcard.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8989	NO_SEARCH_PRIVILEGES
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89FF	NO_FILES_FOUND_ERROR

Remarks

The `searchAttrs` parameter includes system and/or hidden files. If only the system bit is set in the `searchAttrs` parameter, all files are affected except hidden files. If only the hidden bit is set, all files are affected except system files. When neither bit is set (0x00), only files that are not designated either hidden or system are affected.

NOTE: A file is designated hidden or system if its corresponding file attribute is set.

The `searchAttrs` parameter can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x10	\$10	FA_DIRECTORY

The `iterHandle` parameter points to a 9-byte identifier the server uses as an index for searching. In the first call to `NWIntScanFileInformation2`, the first 4 bytes of the number need to be set to `0xFF` accomplished by typecasting the pointer to an `nuint32`, and assigning `-1`, or `0xFFFFFFFF` to it. Every time `NWIntScanFileInformation2` is called, the sequence number for the next iteration is returned.

NCP Calls

0x2222 23 15 Scan File Information

0x2222 23 17 Get File Server Information

0x2222 87 02 Initialize Search

0x2222 87 03 Search For File Or Subdirectory

NWIntScanFileInformation2Ext

Scans the specified directory for the specified file (or directory) and returns the associated directory entry information in the DOS name space, using UTF-8 strings

NetWare Server: 6.5 SP2 or later

Platform: Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWIntScanFileInformation2Ext (
    NWCONN_HANDLE          conn,
    NWDIR_HANDLE           dirHandle,
    const nstr8 N_FAR      *filePattern,
    nuint8                  searchAttrs,
    pnuint8                 iterHandle,
    NW_FILE_INFO2_EXT N_FAR *info,
    nuint16                 augmentFlag);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle relative to the `filePattern` parameter (or 0 if the `filePattern` parameter points to the complete path, including the volume name).

filePattern

(IN) Points to the string containing the file name or wildcard pattern to use in the search. The characters in the string must be UTF-8.

searchAttrs

(IN) Specifies the attributes to use for searching. See Remarks for possible values.

iterHandle

(IN/OUT) Inputs a pointer to the sequence number (set the first 4 bytes to 0xFF initially). Outputs a pointer to the 9-byte sequence number to be used for subsequent iterations.

info

(OUT) Points to the `NW_FILE_INFO2` structure containing the file information.

augmentFlag

(IN) Specifies if wildcards are augmented:

0 = wildcards are not augmented

nonzero = wildcards are augmented

Note that if the high-order bit of a wildcard character is 1, NetWare interprets that character as being a DOS wildcard (which is also called an augmented wildcard) and uses DOS rules for interpretation of that wildcard.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x8989	NO_SEARCH_PRIVILEGES
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89FF	NO_FILES_FOUND_ERROR

Remarks

The `searchAttrs` parameter includes system and/or hidden files. If only the system bit is set in the `searchAttrs` parameter, all files are affected except hidden files. If only the hidden bit is set, all files are affected except system files. When neither bit is set (0x00), only files that are not designated either hidden or system are affected.

NOTE: A file is designated hidden or system if its corresponding file attribute is set.

The `searchAttrs` parameter can have the following values:

C Value	Value Name
0x00	FA_NORMAL
0x02	FA_HIDDEN
0x04	FA_SYSTEM
0x10	FA_DIRECTORY

The `iterHandle` parameter points to a 9-byte identifier the server uses as an index for searching. In the first call to `NWIntScanFileInformation2Ext`, the first 4 bytes of the number need to be set to 0xFF accomplished by typecasting the pointer to an `nuint32`, and assigning -1, or 0xFFFFFFFF to it.

Every time NWIntScanFileInformation2Ext is called, the sequence number for the next iteration is returned. You should not modify this returned sequence number.

NCP Calls

0x2222 23 15 Scan File Information

0x2222 23 17 Get File Server Information

0x2222 87 02 Initialize Search

0x2222 87 03 Search For File Or Subdirectory

0x2222 89 02 Initialize Search

0x2222 89 03 Search For File Or Subdirectory

NWIntScanForTrustees

Scans a directory entry or file for trustees under the specified directory handle and path

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWIntScanForTrustees (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    puint32             iterHandle,
    puint16             numOfEntries,
    NWET_INFO N_FAR    *entryTrusteeInfo,
    nuint16             augmentFlag);
```

Delphi Syntax

```
uses calwin32;

Function NWIntScanForTrustees
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pustr8;
   iterHandle : puint32;
   numOfEntries : puint16;
   Var entryTrusteeInfo : NWET_INFO;
   augmentFlag : nuint16
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle pointing to the directory or file to scan.

path

(IN) Points to an absolute directory or file path (if the `dirHandle` parameter is not specified) or one relative to the `dirHandle` parameter (an absolute path must not be more than 255 bytes long).

iterHandle

(IN/OUT) Points to the server maintained sequence number (set to 0 initially).

numOfEntries

(OUT) Points to the buffer to receive the number of entries returned by `NWIntScanForTrustees`.

entryTrusteeInfo

(OUT) Points to the `NWNET_INFO` structure.

augmentFlag

(IN) Specifies if wildcards are augmented:

- 0 = wildcards are not augmented
- nonzero = wildcards are augmented

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899C	NO_MORE_TRUSTEES

Remarks

`NWIntScanForTrustees` works for both files and directories.

Directories can have any number of objects as trustees. Trustees are returned in groups of 20 `TRUSTEE_INFO` structures. To obtain a complete list, set the `sequence` parameter to 0L for the initial call. `NWIntScanForTrustees` should then be called (for example in a while or do loop) until it returns 0x899C (`NO_MORE_TRUSTEES`). Because 0x899C also indicates `INVALID_PATH`, ensure the `dirHandle/path` parameter combination is correct.

Due to subtle differences in operation, trustees may remain after an iteration, even though not all 20 positions are filled. If a position is not filled, the `objectID` parameter is set to 0L. Check the `objectID` parameter before printing each value in the `objectRights` parameter.

Both the `dirHandle` and `path` parameters must be in the default name space.

The default name space is the name space that matches the OS and the loaded name spaces on that volume. For example, Windows 95 on a volume with LONG name space will set LONG name space as the default name space.

The `dirHandle` parameter can be zero if the `path` parameter points to the complete path, including the volume name. The `path` parameter can point to wildcard characters. However, only the first matching directory is scanned.

NOTE: Call the `NWAllocTemporaryDirectoryHandle` function with the `path` parameter to check for a valid path.

The `NWET_INFO` structure receives trustee information. However, only the `TRUSTEE_INFO` structure is valid for servers 3.x and later. The `sequenceNumber` field should always be ignored.

NCP Calls

0x2222 22 12 Scan Directory For Trustees
0x2222 22 38 Scan File Or Directory For Extended Trustees
0x2222 23 17 Get File Server Information
0x2222 87 05 Scan File Or Subdirectory For Trustees

Example

The following snippet of code shows how to use a do/while loop to repeatedly scan the trustee list for multiple entries. Before displaying the list to a user, the `objectID` and `objectRights` need to be mapped to something easier to read.

```
void PrintTrustees (NWCONN_HANDLE conn, const char *path)
{
    nuint32 iterHandle;
    nuint16 numOfEntries;
    NWET_INFO trusteeInfo;
    NWCCODE ccode;
    int index;

    printf("Trustees for %s:\n", path);
    iterHandle = 0;
    do
    {
        ccode = NWIntScanForTrustees(conn, 0, path, &iterHandle,
&numOfEntries,
        &trusteeInfo, 0);
        if (ccode == NO_MORE_TRUSTEES)
            break;

        if (ccode == 0)
        {
            for (index = 0; index < 20; index++)
            {
                if (trusteeInfo.trusteeList[index].objectID != 0)
                {
                    printf(" 0x%08X: 0x%04X\n",
```

```
        trusteeInfo.trusteeList[index].objectID,  
        trusteeInfo.trusteeList[index].objectRights);  
    }  
}  
}  
} while (ccode == 0);  
}
```

NWIntScanForTrusteesExt

Scans a directory entry or file for trustees of the specified directory handle and path, UTF-8 strings

NetWare Server: 6.5 SP2 or later

Platform: Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWIntScanForTrusteesExt (
    NWCONN_HANDLE          conn,
    NWDIR_HANDLE           dirHandle,
    const nstr8 N_FAR      *path,
    pnuint32                iterHandle,
    pnuint16                numOfEntries,
    NWET_INFO_EXT N_FAR    *entryTrusteeInfo,
    nuint16                 augmentFlag);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle pointing to the directory or file to scan.

path

(IN) Points to an absolute directory or file path (if the `dirHandle` parameter is 0) or one relative to the `dirHandle` parameter. An absolute path must not be more than 255 bytes long. The characters in the string must be UTF-8.

iterHandle

(IN/OUT) Points to the server maintained sequence number (set to 0 initially).

numOfEntries

(OUT) Points to the buffer to receive the number of entries returned by `NWIntScanForTrusteesExt`.

entryTrusteeInfo

(OUT) Points to the `NWNET_INFO_EXT` structure.

augmentFlag

(IN) Specifies if wildcards are augmented:

0 = wildcards are not augmented

nonzero = wildcards are augmented

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899C	NO_MORE_TRUSTEES

Remarks

NWIntScanForTrusteesExt works for both files and directories on NSS volumes.

Directories can have any number of objects as trustees. Trustees are returned in groups of 100 TRUSTEE_INFO structures. To obtain a complete list, set the `iterHandle` parameter to 0 for the initial call. NWIntScanForTrusteesExt should then be called (for example in a while or do loop) until it returns 0x899C (NO_MORE_TRUSTEES). Because 0x899C also indicates INVALID_PATH, ensure the `dirHandle/path` parameter combination is correct.

Due to subtle differences in operation, trustees may remain after an iteration, even though not all 100 positions are filled. If a position is not filled, the `objectID` parameter is set to 0L. Check the `objectID` parameter before printing each value in the `objectRights` parameter.

Both the `dirHandle` and `path` parameters must be in the default name space.

The default name space is the name space that matches the OS and the loaded name spaces on that volume. For example, Windows 95 on a volume with LONG name space will set LONG name space as the default name space.

The `dirHandle` parameter can be zero if the `path` parameter points to the complete path, including the volume name. The `path` parameter can point to wildcard characters. However, only the first matching directory is scanned.

The NWET_INFO_EXT structure receives trustee information. The `sequenceNumber` field should always be ignored.

NCP Calls

0x2222 22 12 Scan Directory For Trustees

0x2222 22 38 Scan File Or Directory For Extended Trustees

0x2222 23 17 Get File Server Information

0x2222 87 05 Scan File Or Subdirectory For Trustees

0x2222 89 05 Scan File Or Subdirectory For Trustees

Example

The following snippet of code shows how to use a do/while loop to repeatedly scan the trustee list for multiple entries. Before displaying the list to a user, the objectID and objectRights need to be mapped to something easier to read.

```
void PrintTrustees (NWCONN_HANDLE conn, const char *path)
{
    nuint32 iterHandle;
    nuint16 numOfEntries;
    NWET_INFO_EXT trusteeInfo;
    NWCCODE ccode;
    int index;

    printf("Trustees for %s:\n", path);
    iterHandle = 0;
    do
    {
        ccode = NWIntScanForTrusteesExt(conn, 0, path, &iterHandle,
            &numOfEntries, &trusteeInfo, 0);
        if (ccode == NO_MORE_TRUSTEES)
            break;

        if (ccode == 0)
        {
            for (index = 0; index < 100; index++)
            {
                if (trusteeInfo.trusteeList[index].objectID != 0)
                {
                    printf(" 0x%08X: 0x%04X\n",
                        trusteeInfo.trusteeList[index].objectID,
                        trusteeInfo.trusteeList[index].objectRights);
                }
            }
        }
    } while (ccode != 0);}
```

NWModifyMaximumRightsMask

Modifies the maximum rights mask of a directory

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWModifyMaximumRightsMask (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              revokeRightsMask,
    nuint8              grantRightsMask);
```

Delphi Syntax

```
uses calwin32

Function NWModifyMaximumRightsMask
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pustr8;
   revokeRightsMask : nuint8;
   grantRightsMask : nuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle for the directory whose maximum rights mask is being modified (or 0 if the `path` parameter points to the complete path, including the volume name).

path

(IN) Points to the absolute directory path (or a path relative to the directory handle) of the directory whose maximum rights mask is being modified.

revokeRightsMask

(IN) Specifies the rights being revoked.

grantRightsMask

(IN) Specifies the rights being granted.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure

Remarks

To modify the maximum rights mask for a directory, the requesting workstation must have access control rights to the directory.

The maximum rights mask follows:

Hex	Bit Definition
0x01	TA_READ
0x02	TA_WRITE
0x08	TA_CREATE
0x10	TA_DELETE
0x20	TA_OWNERSHIP
0x40	TA_SEARCH
0x80	TA_MODIFY

The rights specified by the `revokeRightsMask` parameter are deleted from the maximum rights mask for the directory, and the rights specified by the `grantRightsMask` parameter are added.

The maximum rights mask can be completely reset by setting the `revokeRightsMask` parameter to `0xFF` and then setting the `grantRightsMask` parameter to the desired maximum rights mask. Maximum rights affect the specified directory only and are not inherited by subdirectories.

To return the current rights value, call [NWIntScanDirectoryInformation2 \(page 229\)](#).

NCP Calls

0x2222 22 04 Modify Maximum Rights Mask

0x2222 23 17 Get File Server Information

0x2222 87 07 Modify File or SubDirectory DOS Information

See Also

[NWGetEffectiveRights \(page 200\)](#)

NWRenameDirectory

Renames a NetWare directory

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWRenameDirectory (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *oldName,
    const nstr8 N_FAR  *newName);
```

Delphi Syntax

```
uses calwin32

Function NWRenameDirectory
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   oldName : pstr8;
   newName : pstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle for the directory being deleted (or 0 if the `oldName` parameter points to the complete path, including the volume name).

oldName

(IN) Points to the string containing the name of the directory to be renamed.

newName

(IN) Points to the string containing the new directory name.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8836	NWE_PARAM_INVALID
0x8980	FILE_IN_USE_ERROR
0x898B	NO_RENAME_PRIVILEGES
0x8992	NO_FILES_RENAMED_NAME_EXISTS
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899E	INVALID_FILENAME
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure

Remarks

The `newName` parameter should only include the new name of the directory without listing the volume or directory path. Otherwise, `NWRenameDirectory` will return `NWE_PARAM_INVALID`.

NCP Calls

0x2222 22 15 Rename Directory
0x2222 23 17 Get File Server Information
0x2222 87 04 Rename Or Move A File Or Subdirectory
0x2222 87 22 Generate Directory Base and Volume Number

See Also

[NWCreateDirectory](#) (page 170), [NWDeleteDirectory](#) (page 175)

NWRenameFile

Allows a client to rename a file

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWRenameFile (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       oldDirHandle,
    const nstr8 N_FAR  *oldFileName,
    nuint8              searchAttrs,
    NWDIR_HANDLE       newDirHandle,
    const nstr8 N_FAR  *newFileName);
```

Delphi Syntax

```
uses calwin32

Function NWRenameFile
  (conn : NWCONN_HANDLE;
   oldDirHandle : NWDIR_HANDLE;
   oldFileName : pnstr8;
   searchAttrs : nuint8;
   newDirHandle : NWDIR_HANDLE;
   newFileName : pnstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle containing the file.

oldDirHandle

(IN) Specifies the directory handle containing the file (or 0 if the `oldFileName` parameter points to the complete path, including the volume name).

oldFileName

(IN) Points to a string containing the original name of the file being renamed.

searchAttrs

(IN) Specifies the attributes to use in searching for the specified file.

newDirHandle

(IN) Specifies the new directory handle to contain the specified file.

newFileName

(IN) Points to a string containing the new name of the file.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8987	WILD_CARDS_IN_CREATE_FILE_NAME or CREATE_FILENAME_ERROR
0x898B	NO_RENAME_PRIVILEGES
0x898D	SOME_FILES_AFFECTED_IN_USE
0x898E	NO_FILES_AFFECTED_IN_USE
0x898F	SOME_FILES_AFFECTED_READ_ONLY
0x8990	NO_FILES_AFFECTED_READ_ONLY
0x8991	SOME_FILES_RENAMED_NAME_EXISTS
0x8992	NO_FILES_RENAMED_NAME_EXISTS
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899A	RENAMING_ACROSS_VOLUMES
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	NO_FILES_FOUND_ERROR

Remarks

The source directory (where the file resides) and the target directory (where the renamed file is to be deposited) do not need to be the same directory. However, the two files must reside on the same server. `NWRenameFile` cannot move a file from one server to another or from one volume to another.

The `searchAttrs` parameter is used to include system and/or hidden files. If only the system bit is set in the `searchAttrs` parameter, all files are affected except hidden files. If only the hidden

bit is set, all files are affected except system files. When neither bit is set (0x00), only files that are not designated either hidden or system are affected.

NOTE: A file is designated hidden or system if its corresponding file attribute is set.

The `searchAttrs` parameter can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x01	\$01	FA_READ_ONLY
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x08	\$08	FA_EXECUTE_ONLY
0x10	\$10	FA_DIRECTORY
0x20	\$20	FA_NEEDS_ARCHIVED
0x80	\$80	FA_SHAREABLE

Since the path length is restricted to 256 bytes, applications must call the `NWAllocTemporaryDirectoryHandle` function to allocate the `dirHandle` parameter for path lengths greater than 256 bytes.

NCP Calls

0x2222 23 17 Get File Server Information
0x2222 69 Rename File
0x2222 87 04 Rename Or Move A File Or Subdirectory

See Also

[NWAllocTemporaryDirectoryHandle \(page 163\)](#)

10.5 NWS*-NWZ* Functions

Click on any function name in the table of contents to view the purpose, syntax, parameters, and return values for that function.

- ◆ [“NWScanConnectionsUsingFile” on page 260](#)
- ◆ [“NWScanDirectoryForTrustees2” on page 262](#)
- ◆ [“NWScanOpenFilesByConn2” on page 265](#)
- ◆ [“NWSetCompressedFileLengths” on page 267](#)
- ◆ [“NWSetCompressedFileSize” on page 269](#)
- ◆ [“NWSetDirectoryHandlePath” on page 271](#)
- ◆ [“NWSetDirectoryInformation” on page 274](#)

- ◆ “NWSetDirEntryInfo” on page 277
- ◆ “NWSetDirSpaceLimit” on page 281
- ◆ “NWSetExtendedFileAttributes2” on page 283
- ◆ “NWSetFileAttributes” on page 286
- ◆ “NWSetFileInformation2” on page 289
- ◆ “NWSetVolumeFlags” on page 292
- ◆ “NWVolumeIsCDROM” on page 294

NWScanConnectionsUsingFile

Scans all connections using a specified file

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWScanConnectionsUsingFile (
    NWCONN_HANDLE          conn,
    NWDIR_HANDLE           dirHandle,
    const nstr8 N_FAR      *filePath,
    pnint16                 iterHandle,
    CONN_USING_FILE N_FAR  *fileUse,
    CONNS_USING_FILE N_FAR *fileUsed);
```

Delphi Syntax

```
uses calwin32

Function NWScanConnectionsUsingFile
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   filePath : pustr8;
   iterhandle : pnint16;
   Var fileUse : CONN_USING_FILE;
   Var fileUsed : CONNS_USING_FILE
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired directory path. Use the DOS namespace as input parameter for the full path of filename, when the directory handle is 0.

filePath

(IN) Points to a full file path (or a path relative to `dirHandle`) specifying the file to be checked (wildcards are not allowed).

iterHnd

(IN/OUT) Points to the next record to be scanned (0 initially).

fileUse

(OUT) Points to the CONN_USING_FILE structure.

fileUsed

(OUT) Points to the CONNS_USING_FILE structure.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88FF	NWE_REQUESTER_FAILURE: Scan Completed
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A8	ERR_ACCESS_DENIED
0x89C6	NO_CONSOLE_PRIVILEGES

Remarks

You must have console operator rights to call NWScanConnectionsUsingFile.

Upon each subsequent call, the number of the next record to be scanned is returned in the `iterHnd` parameter. This value should not be changed during the scan. NWScanConnectionsUsingFile returns 0xFFFFFFFF upon completion.

If no connections are using the specified file, the structure returned by the `fileUsed` parameter will contain zeroes. Check the `connCount` parameter in the returned structure to see the number of connections actually using the file.

If the `fileUse` parameter is NULL, the records are returned in the `fileUsed` parameter in groups, instead of one at a time.

Use the DOS namespace as input parameter for the full path of filename, when the `directoryhandle` is 0.

NCP Calls

- 0x2222 23 17 Get File Server Information
- 0x2222 23 236 Get Connections Using A File
- 0x2222 23 244 Convert Path To Dir Entry

NWScanDirectoryForTrustees2

Scans a directory for trustees using the specified path and directory handle

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWScanDirectoryForTrustees2 (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *srchPath,
    puint32             iterHandle,
    pustr8              dirName,
    puint32             dirDateTime,
    puint32             ownerID,
    TRUSTEE_INFO N_FAR *trusteeList);
```

Delphi Syntax

```
uses calwin32;

Function NWScanDirectoryForTrustees2
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   searchPath : pustr8;
   iterHandle : puint32;
   dirName : pustr8;
   dirDateTime : puint32;
   ownerID : puint32;
   Var trusteeList : TRUSTEE_INFO
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle for the directory being scanned (0 if the `srchPath` parameter points to the complete path, including the volume name).

srchPath

(IN) Points to an absolute directory path (or a path relative to the directory handle) and a search pattern.

iterHandle

(IN/OUT) Points to the sequence number to be used for subsequent calls (0 initially).

dirName

(OUT) Points to the directory name found (optional, up to 256 bytes).

dirDateTime

(OUT) Points to the creation date and time of the directory (optional).

ownerID

(OUT) Points to the object ID of the directory owner (optional).

trusteeList

(OUT) Points to an array of 20 TRUSTEE_INFO structures.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	NO_MORE_TRUSTEES

Remarks

The `srchPath` parameter can include wildcard characters.

Directories can have any number of objects as trustees. The directory trustees are stored and retrieved in groups on the server. To obtain a complete list, use the `iterHandle` parameter.

`NWScanDirectoryForTrustees2` increments the value referenced by the `iterHandle` parameter to the next appropriate value. For subsequent calls, pass in the new value of the `iterHandle` parameter.

Trustees are returned in groups of 20 TRUSTEE_INFO structures. Due to subtle differences in operation, trustees may remain after an iteration, even though not all 20 positions are filled. If a position is not filled, the `objectID` field of TRUSTEE_INFO points to a value of 0L.

NWScanDirectoryForTrustees2 should be called until it returns 0x899C (NO_MORE_TRUSTEES). Because 0x899C also means INVALID_PATH, ensure the dirHandle/pbstrSrchPath parameter combination is correct.

NULL can be substituted for all optional items. However, all parameter positions must be filled.

NCP Calls

0x2222 22 1 Get Directory Path

0x2222 22 2 Scan Directory Information

0x2222 22 12 Scan Directory For Trustees

0x2222 22 38 Trustees Scan Ext

0x2222 23 17 Get File Server Information

0x2222 87 02 Initialize Search

0x2222 87 03 Search For File or Subdirectory

0x2222 87 05 Scan File Or Subdirectory For Trustees

0x2222 87 06 Obtain File or Subdirectory Information

See Also

[NWScanNSDirectoryForTrustees \(page 530\)](#)

NWScanOpenFilesByConn2

Scans information about the files opened by a specified connection

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWScanOpenFilesByConn2 (
    NWCONN_HANDLE          conn,
    NWCONN_NUM             connNum,
    puint16                iterHandle,
    OPEN_FILE_CONN_CTRL N_FAR *openCtrl,
    OPEN_FILE_CONN N_FAR   *openFile);
```

Delphi Syntax

```
uses calwin32

Function NWScanOpenFilesByConn2
  (conn : NWCONN_HANDLE;
   connNum : NWCONN_NUM;
   iterHandle : puint16;
   Var openCtrl : OPEN_FILE_CONN_CTRL;
   Var openFile : OPEN_FILE_CONN
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

connNum

(IN) Specifies the connection number of the logged-in object to be scanned.

iterHandle

(IN/OUT) Points to the next record to be scanned (0 initially).

openCtrl

(OUT) Points to the OPEN_FILE_CONN_CTRL structure.

openFile

(OUT) Points to the OPEN_FILE_CONN structure.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88FF	Scan Completed
0x89FD	BAD_STATION_NUMBER

Remarks

For 3.x, you must have console operator rights to call NWScanOpenFilesByConn2 or NO_CONSOLE_PRIVILEGES will be returned.

For 4.x, 5.x, and 6.x, you can call NWScanOpenFilesByConn2 to return information about the connection without needing console operator privileges. To return information about other connection numbers, you must have console rights. A client with console privileges can pass any valid connection number to NWScanOpenFilesByConn2 and receive information about that connection.

Upon each subsequent call, the `iterHandle` parameter returns the number of the next record to be scanned and points to 0xFFFFFFFF upon completion. It should not be changed during the scan.

The OPEN_FILE_CONN_CTRL structure is used internally and should not be written to.

NCP Calls

0x2222 23 17 Get File Server Information

0x2222 23 235 Get Connection's Open Files

See Also

[NWGetPathFromDirectoryBase \(page 620\)](#)

NWSetCompressedFileLengths

Sets the uncompressed and compressed lengths of a file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfinfo.h>

int NWSetCompressedFileLengths (
    int    handle,
    LONG   uncompressedLength,
    LONG   compressedLength ;
```

Parameters

handle

(IN) Specifies the handle of the file for which to set the lengths.

uncompressedLength

(IN) Specifies the length of the file in an uncompressed state.

compressedLength

(IN) Specifies the length of the file after being compressed.

Return Values

0x00	Success
0xFF	Failure

Remarks

NWSetCompressedFileLengths sets the compressed and uncompressed lengths of a file.

NWSetCompressedFileLengths is useful for restoring directory entry information about files that have previously been backed up.

The `uncompressedLength` parameter is the length normally seen in normal directory listings.

See Also

[NWGetCompressedFileLengths \(page 186\)](#)

NWSetCompressedFileSize

Attempts to set the logical file size for a compressed file

NetWare Server: 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWSetCompressedFileSize (
    NWCONN_HANDLE   conn,
    nuint32          fileHandle,
    nuint32          reqFileSize,
    pnuint32         resFileSize);
```

Delphi Syntax

```
uses calwin32

Function NWSetCompressedFileSize
  (conn : NWCONN_HANDLE;
   fileHandle : nuint32;
   reqFileSize : nuint32;
   resFileSize : pnuint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the connection handle of the associated NetWare server.

fileHandle

(IN) Specifies an OS or NetWare file handle.

reqFileSize

(IN) Specifies the requested file size.

resFileSize

(OUT) Points to the size actually assigned by the OS.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8988	INVALID_FILE_HANDLE
0x89A8	ERR_ACCESS_DENIED

Remarks

The logical file size is the true size of the file as reported by the client operating systems. When a file is compressed, it shrinks in physical size. However, its logical size should remain the same. In cases where the client forces the creation of a compressed file (by opening a file in compressed mode), the NetWare OS gets the actual size of the file by calling `NWSetCompressedFileSize`.

If the `fileHandle` parameter contains a NetWare handle, the `conn` parameter contains the connection handle of the associated server. If NETX is running and a DOS file handle is passed, the `conn` parameter must also contain a valid connection ID. In all other circumstances, the `conn` parameter is ignored.

NCP Calls

0x2222 90 12 Set Compressed File Size

NWSetDirectoryHandlePath

Sets the target directory handle for the specified directory handle and path

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWSetDirectoryHandlePath (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       sourceDirHandle,
    const nstr8 N_FAR  *dirPath,
    NWDIR_HANDLE       destDirHandle);
```

Delphi Syntax

```
uses calwin32;

Function NWSetDirectoryHandlePath
  (conn : NWCONN_HANDLE;
   sourceDirHandle : NWDIR_HANDLE;
   dirPath : pnstr8;
   destDirHandle : NWDIR_HANDLE
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

sourceDirHandle

(IN) Specifies the source directory handle (index number) identifying the volume or directory on a NetWare server being reassigned (1-255).

dirPath

(IN) Points to the source directory path (optional).

destDirHandle

(IN) Specifies the target directory handle (index number) to become the new directory handle for the specified directory.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FA	TEMP_REMAP_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure

Remarks

If `NWSetDirectoryHandlePath` fails, the `destDirHandle` parameter remains unchanged.

In cases where multiple NetWare servers are being used, the `sourceDirHandle` and `destDirHandle` parameters must have the same server connection handle identifier.

`NWSetDirectoryHandlePath` assigns the `destDirHandle` parameter to a directory path defined by combining the `sourceDirHandle` parameter and the string accessed by the `dirPath` parameter.

A NetWare server maintains a Directory Handle Table for each workstation that is logged in.

The `destDirHandle` parameter is another index number from the Directory Handle Table for the NetWare server.

The `dirPath` parameter can identify a full or partial directory path. A full directory path defines a volume or a directory on a given NetWare server in the format `VOLUME:DIRECTORY/.../ DIRECTORY`. A partial directory path specifies at least a directory and one or more parent directories.

Applications frequently combine a directory handle and a directory path to specify a target directory. For example, if the specified directory handle points to `SYS:` and the specified directory path is `PUBLIC/WORDP`, the specified directory is `SYS:PUBLIC/WORDP`.

When an application defines a target directory using only a directory handle, the application must set the `dirPath` parameter to a NULL string. When an application defines a directory using only a directory path, the application must set the `sourceDirHandle` parameter to zero.

NCP Calls

0x2222 22 00 Set Directory Handle

0x2222 23 17 Get File Server Information

0x2222 87 09 Set Short Directory Handle

See Also

[NWGetDirectoryHandlePath \(page 191\)](#)

NWSetDirectoryInformation

Changes information about a directory including the creation date and time, owner object ID, and maximum rights mask

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWSetDirectoryInformation (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint32             dirDateTime,
    nuint32             ownerID,
    nuint8              rightsMask);
```

Delphi Syntax

```
uses calwin32

Function NWSetDirectoryInformation
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   path : pustr8;
   dirDateTime : nuint32;
   ownerID : nuint32;
   rightsMask : nuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle (index number from 1-255) pointing to the directory, partial directory, or volume whose information is being set (0 if the `path` parameter points to the complete path, including the volume name).

path

(IN) Points to the directory path of the directory being changed.

dirDateTime

(IN) Specifies the new creation date and time.

ownerID

(IN) Specifies the object ID of the owner who created the directory.

rightsMask

(IN) Specifies the new maximum rights mask for the directory.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89F0	WILD_CARD_NOT_ALLOWED
0x89FF	Failure, NO_FILES_FOUND_ERROR

Remarks

NWSetDirectoryInformation defines the target directory by passing a directory handle and a directory path.

A NetWare server maintains a Directory Handle Table for each logged in workstation.

The `path` parameter cannot contain wild card characters or NWSetDirectoryInformation will return WILD_CARD_NOT_ALLOWED.

The `path` parameter can identify a full or partial directory path. A full directory path defines a volume or a directory on a given NetWare server in the format VOLUME:DIRECTORY/.../ DIRECTORY. A partial directory path specifies at least a directory, and possibly one or more parent directories.

Applications frequently combine a directory handle and a directory path to specify a target directory. For example, if the specified directory handle points to SYS: and the specified directory path is PUBLIC/WORDP, the specified directory is SYS:PUBLIC/WORDP.

The `dirDateTime` parameter appears in standard DOS format. The first two bytes contain the year (7 bits), month (4 bits), and day (5 bits) fields, and the second two bytes contain the hour (5 bits), minute (6 bits), and second (5 bits) fields.

NWSetDirectoryInformation sets the date and time in ascending order (byte 1, byte 2, byte 3, byte 4). The date and time values are defined as follows:

Type	Value
Year	0=1980, 1=1981, ..., 119=2099
Month	1 to 12
Day	1 to 31
Hour	0 to 23
Minute	0 to 59
Second	0 to 29 (in units of 2 seconds)

The `rightsMask` parameter contains the maximum rights mask for the subdirectory. The bits in the maximum rights mask are defined as follows:

0x00 = TA_NONE
0x01 = TA_READ
0x02 = TA_WRITE
0x04 = TA_OPEN
0x08 = TA_CREATE
0x10 = TA_DELETE
0x20 = TA_OWNERSHIP
0x40 = TA_SEARCH
0x80 = TA_MODIFY
0xFB = TA_ALL

NOTE: TA_OPEN is obsolete in version 3.x and above.

To change information for a directory, the requesting workstation must have access control rights and modify rights to the directory's parent. Only a workstation with SUPERVISOR rights can change the owner of a directory.

NCP Calls

0x2222 22 25 Set Directory Information

0x2222 23 17 Get File Server Information

0x2222 87 07 Modify File Or Subdirectory DOS Information

See Also

[NWParseNetWarePath \(page 622\)](#)

NWSetDirEntryInfo

Changes information about a directory entry (file or directory)

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwentry.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWSetDirEntryInfo (
    NWCONN_HANDLE          conn,
    NWDIR_HANDLE           dirHandle,
    nuint8                  searchAttrs,
    nuint32                  iterHandle,
    nuint32                  changeBits,
    const NWENTRY_INFO N_FAR *newEntryInfo);
```

Delphi Syntax

```
uses calwin32

Function NWSetDirEntryInfo
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   searchAttrs : nuint8;
   iterHandle : nuint32;
   changeBits : nuint32;
   Var newEntryInfo : NWENTRY_INFO
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle.

searchAttrs

(IN) Specifies the search attribute to use in searching for the directory entry.

iterHandle

(IN) Is currently unused and ignored for NetWare 3.11 and later. For NetWare versions prior to 3.11, it can be used iteratively to find all files that match a specified search criteria.

changeBits

(IN) Specifies the set of bits to indicate which attributes to change.

newEntryInfo

(IN) Points to the NWENTRY_INFO structure.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

NWSetDirEntryInfo only works with 3.11 and above servers.

For files, the `dirHandle` parameter must point to parent directory. For directories, it should follow the same conventions as for the `NWIntScanDirEntryInfo` function.

The `searchAttrs` parameter specifies the kind of entry to look for (hidden, system, etc.). For example, if only the system bit is set in the `searchAttrs` parameter, all files except hidden files are affected. If only the hidden bit is set, all files except system files are affected. If neither bit is set (0x00), only files not designated either hidden or system are affected. On NetWare versions previous to 3.11, you might need to use `iterHandle` to call this function iteratively to eventually affect all files that fit a particular search attribute since `NWSetDirEntryInfo` affects only one file or directory at a time.

NOTE: A file is designated hidden or system if its corresponding file attribute is set.

`searchAttrs` can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x10	\$10	FA_DIRECTORY

changeBits can have the following values:

C Value	Delphi Value	Value Name
0x0001L	\$0001	MModifyNameBit
0x0002L	\$0002	MFileAttributesBit
0x0004L	\$0004	MCreateDateBit
0x0008L	\$0008	MCreateTimeBit
0x0010L	\$0010	MOwnerIDBit
0x0020L	\$0020	MLastArchivedDateBit
0x0040L	\$0040	MLastArchivedTimeBit
0x0080L	\$0080	MLastArchivedIDBit
0x0100L	\$0100	MLastUpdatedDateBit
0x0200L	\$0200	MLastUpdatedTimeBit
0x0400L	\$0400	MLastUpdatedIDBit
0x0800L	\$0800	MLastAccessedDateBit
0x1000L	\$1000	MInheritedRightsMaskBit
0x2000L	\$2000	MMaximumSpaceBit

The `NWENTRY_INFO` structure must be initialized to 0 before calling the `NWSetDirEntryInfo` function.

To change information for a directory, the requesting workstation must have access control and modify rights. Only a workstation with `SUPERVISOR` rights can change the owner of a directory. The `lastModifyDateAndTime` field in the `NWDIR_INFO` structure cannot be changed for volumes. Otherwise, the last modified date and time will be set to the current date and time.

For files, the `dirHandle` parameter must point to the parent directory. The `nameLength` and `name` fields in the `NWENTRY_INFO` structure must contain the specific file information.

For directories, if the `dirHandle` parameter points to the parent directory, the `nameLength` and `name` fields in the `NWENTRY_INFO` structure must contain the specific directory information.

For directories, if the `dirHandle` parameter points to the specific directory itself, the `nameLength` field must be set to 0.

For each name space, the `dirHandle` parameter and the `nameSpace`, `name` and `nameLength` fields must be synchronized to indicate the correct name space.

NCP Calls

0x2222 22 37 Set Directory Entry Information

0x2222 23 17 Get File Server Information

0x2222 87 07 Modify File Or Subdirectory DOS Information

See Also

[NWIntScanDirEntryInfo \(page 232\)](#), [NWSetNSEntryDOSInfo \(page 551\)](#)

NWSetDirSpaceLimit

Specifies a space limit (in 4 KB blocks) on a particular subdirectory

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwdirect.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWSetDirSpaceLimit (
    NWCONN_HANDLE   conn,
    NWDIR_HANDLE    dirHandle,
    nuint32          spaceLimit);
```

Delphi Syntax

```
uses calwin32;

Function NWSetDirSpaceLimit
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   spaceLimit : nuint32
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the NetWare directory handle pointing to the directory to scan.

spaceLimit

(IN) Specifies the directory space limit (in 4 KB sizes).

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION

0x8901	ERR_INSUFFICIENT_SPACE
0x898C	NO_MODIFY_PRIVILEGES
0x89BF	INVALID_NAME_SPACE

Remarks

If the space limit is set to 0, space limit restrictions are lifted. If the restriction is 0xFFFFFFFF, the space limit on the directory is set to 0.

Before space limit restrictions can be lifted, they must previously have been set. If 0 is passed to the `spaceLimit` parameter when no restrictions are set, `NWSetDirSpaceLimit` fails and returns 0x89FF.

NOTE: All restrictions are set in units of 4K blocks.

NSS volumes and traditional volumes have very different architectures, so this function behaves differently, depending upon the volume the directory resides on. For example, traditional volumes take a long time to mount because as the volume mounts, all entries are placed in memory and disk space usage information is calculated and kept current. NSS volumes mount quickly because the entire file system is not scanned and thus disk space usage information must be calculated when a request comes in. For a few disk space requests, you will not see a great deal of difference between an NSS volume and a traditional volume. However, if you send through 3000 requests at the same time to an NSS volume, utilization can spike to 100%, causing the server to drop connections.

NCP Calls

0x2222 22 36 Set Directory Disk Space Restrictions

See Also

[NWGetDirSpaceLimitList \(page 195\)](#), [NWGetDirSpaceLimitList2 \(page 197\)](#)

NWSetExtendedFileAttributes2

Sets the extended attributes of a file

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWSetExtendedFileAttributes2 (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              extAttrs);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWSetExtendedFileAttributes2
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   path : pnstr8;
   extAttrs : nuint8
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the connection handle.

dirHandle

(IN) Specifies the directory handle of the root directory of the new directory..

path

(IN) Points to the string containing the name and path of the new directory.

extAttrs

(IN) Specifies the extended attributes for the file.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x898D	SOME_FILES_AFFECTED_IN_USE
0x898E	NO_FILES_AFFECTED_IN_USE
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure, NO_FILES_FOUND_ERROR

Remarks

NWSetExtendedFileAttributes2 requires Search rights to the directory where the file resides.

The `path` parameter can specify either the complete path name for a file or a path relative to the current working directory.

For example, if the complete path name is `SYS:ACCOUNT/DOMEST/TARGET.DAT` and the directory handle mapping is `SYS:ACCOUNT`, the `path` parameter could point to either of the following:

`SYS:/ACCOUNT/DOMEST/TARGET.DAT` or `DOMEST/TARGET.DAT`

The bit map for the `extAttrs` parameter follows:

0-2	Search mode bits
4	Transaction bit
6	Read audit bit (not yet implemented)
7	Write audit bit (not yet implemented)

Setting the transaction bit prompts TTS™ to track all Writes to the file during a transaction. A transaction file cannot be deleted or renamed until the transaction bit is turned off by calling `NWSetExtendedFileAttributes2`.

Setting the index bit prompts NetWare to index the File Allocation Tables for the file, thereby reducing the time required to access files. Files larger than 2 MB should have this bit set.

NOTE: To modify further extended file attributes, use [NWSetNSEntryDOSInfo \(page 551\)](#).

NCP Calls

0x2222 79 Set File Extended Attribute

See Also

[NWGetExtendedFileAttributes2 \(page 206\)](#), [NWSetNSEntryDOSInfo \(page 551\)](#)

NWSetFileAttributes

Modifies a file's original attributes

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWSetFileAttributes (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *fileName,
    nuint8              searchAttrs,
    nuint8              newAttrs);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWSetFileAttributes
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   fileName : pustr8;
   searchAttrs : nuint8;
   newAttrs : nuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle containing the file.

dirHandle

(IN) Specifies the NetWare directory handle (0 if the `fileName` parameter points to the complete path, including the volume name).

fileName

(IN) Points to the string containing a path name, relative to `dirHandle`.

searchAttrs

(IN) Specifies the attributes to use in searching for a file.

newAttrs

(IN) Specifies the new attributes to be applied to the file designated by the `dirHandle` and `fileName` parameters.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x898D	SOME_FILES_AFFECTED_IN_USE
0x898E	NO_FILES_AFFECTED_IN_USE
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure, NO_FILES_FOUND_ERROR

Remarks

The `fileName` parameter can specify either a complete path name or a path relative to the current working directory. For example, if the complete path name is `SYS:ACCOUNT/DOMEST/TARGET.DAT` and the directory handle mapping is `SYS:ACCOUNT`, the `fileName` parameter could point to either of the following:

```
SYS:ACCOUNT/DOMEST/TARGET.DAT or  
DOMEST/TARGET.DAT
```

The `searchAttrs` parameter includes system and/or hidden files. If only the system bit is set in the `searchAttrs` parameter, all files are affected except hidden files. If only the hidden bit is set, all files are affected except system files. When neither bit is set (0x00), only files that are not designated either hidden or system are affected.

NOTE: A file is designated hidden or system if its corresponding file attribute is set.

The `searchAttrs` parameter can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x01	\$01	FA_READ_ONLY

C Value	Delphi Value	Value Name
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x08	\$08	FA_EXECUTE_ONLY
0x10	\$10	FA_DIRECTORY
0x20	\$20	FA_NEEDS_ARCHIVED
0x80	\$80	FA_SHAREABLE

NCP Calls

0x2222 23 17 Get File Server Information

0x2222 70 Set File Attributes

0x2222 87 07 Modify File Or Subdirectory DOS Information

See Also

[NWGetExtendedFileAttributes2 \(page 206\)](#), [NWIntScanFileInformation2 \(page 238\)](#),
[NWSetFileInformation2 \(page 289\)](#), [NWSetNSEntryDOSInfo \(page 551\)](#)

NWSetFileInformation2

Updates file information

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWSetFileInformation2 (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *fileName,
    nuint8              searchAttrs,
    NW_FILE_INFO2 N_FAR *info);
```

Delphi Syntax

```
uses calwin32

Function NWSetFileInformation2
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   fileName : pnstr8;
   searchAttrs : nuint8;
   Var info : NW_FILE_INFO2
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle containing the file to be modified.

dirHandle

(IN) Specifies the NetWare directory handle (0 if the `fileName` parameter points to the complete path, including the volume name).

fileName

(IN) Points to the name of the file to modify. The name, or complete path, must be in the long name space to work on Windows workstations.

searchAttrs

(IN) Specifies the search attributes.

info

(IN) Points to NW_FILE_INFO2.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8988	INVALID_FILE_HANDLE
0x898C	NO_MODIFY_PRIVILEGES
0x898E	NO_FILES_AFFECTED_IN_USE
0x8994	NO_WRITE_PRIVILEGES_OR_READONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89A2	READ_FILE_WITH_RECORD_LOCKED
0x89FC	NO_SUCH_OBJECT
0x89FD	BAD_STATION_NUMBER
0x89FE	DIRECTORY_LOCKED
0x89FF	Failure, NO_FILES_FOUND_ERROR

Remarks

NWSetFileInformation2 handles long names (up to 256 bytes).

NWSetFileInformation2 sets the file information defined by the NW_FILE_INFO2 structure.

The `fileName` parameter can specify either a complete path name or a path relative to the current working directory. For example, if the complete path name is SYS:ACCOUNT/DOMEST/TARGET.DAT, and the directory handle mapping is SYS:ACCOUNT, the `fileName` parameter could be either of the following:

```
SYS:ACCOUNT/DOMEST/TARGET.DAT or  
DOMEST/TARGET.DAT
```

The `searchAttrs` parameter is used to include system and/or hidden files. If only the system bit is set in the `searchAttrs` parameter, all files are affected except hidden files. If only the hidden bit is set, all files are affected except system files. When neither bit is set (0x00), only files that are not designated hidden or system are affected.

NOTE: A file is designated hidden or system if its corresponding file attribute is set.

The `searchAttrs` parameter can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x01	\$01	FA_READ_ONLY
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x08	\$08	FA_EXECUTE_ONLY
0x10	\$10	FA_DIRECTORY
0x20	\$20	FA_NEEDS_ARCHIVED
0x80	\$80	FA_SHAREABLE

NCP Calls

0x2222 23 16 Set File Information

0x2222 23 17 Get File Server Information

0x2222 87 07 Modify File Or Subdirectory DOS Information

See Also

[NWGetExtendedFileAttributes2 \(page 206\)](#), [NWIntScanFileInformation2 \(page 238\)](#), [NWSetFileAttributes \(page 286\)](#),

NWSetVolumeFlags

Sets the specified flags on a volume

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfileio.h>
```

```
LONG NWSetVolumeFlags (
    LONG    volume,
    LONG    flags);
```

Parameters

volume

(IN) Specifies the volume to set attributes on.

flags

(IN) Specifies the flags to set for the specified volume.

Return Values

0	Success
---	---------

-1	Failure
----	---------

Remarks

`flags` can have the following values:

0x02	SUB_ALLOCATION_FLAG: If set, sub allocation units are valid on this volume.
0x04	FILE_COMPRESSION_FLAGS: If set, file compression is enabled on this volume.
0x08	DATA_MIGRATION_FLAG: If set, data migration is allowed on this volume.
0x40	VOLUME_IMMEDIATE_PURGE_FLAG: If set, this volume's deleted files will be purged immediately.

See Also

[NWGetVolumeFlags \(page 216\)](#)

NWVolumeIsCDROM

Determines whether a given volume is a CD-ROM or a read-only volume

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwdir.h>

int NWVolumeIsCDROM (
    LONG    volumeNumber,
    LONG    *isCDROM ;
```

Parameters

volumeNumber

(IN) Specifies the number of the volume to be queried.

isCDROM

(OUT) Points to either TRUE or FALSE, indicating whether the volume is a CD-ROM volume for NetWare 4.x or a read-only volume for NetWare 5.x and 6.x.

Return Values

0	ESUCCESS
0xFFFF	Failure—NWErrno is set with the appropriate error code.

Remarks

NWVolumeIsCDROM allows you to determine if the given volume is a CD-ROM volume (for NetWare 4.x) and if it is a read-only volume for NetWare 5. All CD-ROM volumes are also read-only volumes.

NWVolumeIsCDROM fails if the given volume is not mounted.

See Also

[NWGetExtendedVolumeInfo](#), [NWGetVolumeName](#) (Volume Management)

10.6 O*-Z* Functions

Click on any function name in the table of contents to view the purpose, syntax, parameters, and return values for that function.

- ♦ [“opendir” on page 296](#)
- ♦ [“PurgeErasedFile” on page 298](#)
- ♦ [“readdir” on page 300](#)
- ♦ [“remove” on page 302](#)
- ♦ [“rename” on page 304](#)
- ♦ [“rmdir” on page 306](#)
- ♦ [“SalvageErasedFile” on page 307](#)
- ♦ [“ScanErasedFiles” on page 309](#)
- ♦ [“SetExtendedFileAttributes” on page 311](#)
- ♦ [“SetFileInfo” on page 313](#)
- ♦ [“SetReaddirAttribute” on page 316](#)
- ♦ [“_splitpath” on page 318](#)
- ♦ [“stat” on page 320](#)
- ♦ [“tmpnam” on page 322](#)
- ♦ [“umask” on page 323](#)
- ♦ [“UnAugmentAsterisk” on page 324](#)
- ♦ [“unlink” on page 325](#)
- ♦ [“UseAccurateCaseForPaths” on page 326](#)
- ♦ [“utime” on page 327](#)

opendir

Opens a directory for reading with the attributes set by calling `SetReaddirAttribute` and the next matching file found by calling `readdir` functions

Local Servers: blocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <dirent.h>
```

```
DIR * opendir (
    const char *pathname);
```

Parameters

pathname

(IN) Can be either relative to the current working directory or it can be an absolute path name (must include file specification—accepts wild cards).

Return Values

Returns a pointer to the `DIR` structure (required for subsequent calls to the `readdir` function) containing the file names matching the pattern specified by the `pathname` parameter.

Returns `NULL` if the path name is not valid or if there are no files matching the path name. If an error occurs, `errno` and `NetWareErrno` are set.

Remarks

The last part of the path name can contain the characters `??` and `*` for matching multiple files, as in the following example:

```
odir = opendir ("sys:\\public\\*.");
```

More than one directory can be read at the same time by calling the `opendir`, `readdir`, and `closedir` functions.

`opendir` calls the `malloc` function to allocate memory for a `DIR` structure. The `closedir` function frees the memory.

Information about the first file or directory matching the specified path name is not placed in the `DIR` structure until after the first call to the `readdir` function.

Beginning with Release 9 of NW SDK, `opendir` returns long names in the `d_name` field of the `dirent` structure if the target namespace is previously set to something other than DOS by calling

SetTargetNameSpace. To have use of this long name functionality, you must compile with the dirent.h file included with Release 9 or later. In addition, with NetWare versions lower than 5, you might need CLIBAUX.NLM loaded on the server for symbol resolution. (Currently opendir support for spaces other than DOS is available only on calls to the local server.)

NOTE: The position in the structure of the `d_name` field prior to Release 9 has been assumed by the new `d_nameDOS` field to ensure backward compatibility, and the `d_name` field has been moved to the end of the structure. The new code puts the DOS name space name at the `d_nameDOS` field offset so old code will still work. This can all be done with relative ease because CLIB allocates the memory.

By default, opendir returns all file and directory names when the pattern `*.*` is specified for the DOS name space only (only names with one dot are returned for the long name space). To use `*.*` to return all names for the long name space, call `UnAugmentAsterisk` before calling opendir. You can also call `SetCurrentNameSpace(0)` to set the name space to DOS, call opendir, then call `SetCurrentNameSpace(4)` to reset the name space to long.

See Also

[closedir \(page 143\)](#), [readdir \(page 300\)](#), [SetReaddirAttribute \(page 316\)](#), [UnAugmentAsterisk \(page 324\)](#)

PurgeErasedFile

Permanently deletes a file that has been marked for deletion

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfinfo.h>

int PurgeErasedFile (
    char *pathname,
    long  sequenceNumber);
```

Parameters

pathname

(IN) Specifies the string containing either the absolute path (including the volume name) or the relative path name of the file to purge (maximum 255 characters, including the NULL terminator).

sequenceNumber

(IN) Identifies which version of the specified file to purge.

Return Values

0	0x00	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

An application marks a file for deletion with the `remove` or `unlink` function. However, the server does not permanently delete a file until the server needs the disk space occupied by the file marked for deletion. A file marked for deletion with the `remove` or `unlink` functions can be recovered by calling the `SalvageErasedFile` function.

`PurgeErasedFile` permanently deletes a file marked for deletion. It frees the disk space that the deleted file occupied. A file deleted with `PurgeErasedFile` cannot be recovered.

NOTE: The `sequenceNumber` parameter must be obtained from the `ScanErasedFiles` function. The current connection must have Delete rights to the file.

There is no need to call the `ScanErasedFiles` function to get a sequence number on remote 286 servers. `PurgeErasedFile` can be called without regard to the validity of the path name or sequence number on 286 servers. 286 servers do not retain files that have been marked for deletion but not yet purged. `PurgeErasedFile` purges all files that have been deleted recently (since the last file system operation).

The `SetCurrentNameSpace` function sets the name space that is used for parsing the path input to `PurgeErasedFile`.

NOTE: `PurgeErasedFile` currently works only in the DOS name space. However, you can purge a file in other name spaces in the following way. Call the `SetCurrentNameSpace` function to change to the DOS name space and then call the `ScanErasedFiles` function to get the DOS names of the files you want to purge. These are returned in the structure that the `ScanErasedFiles` function uses. You can then purge the files, supplying their DOS names as specified by the `pathname` parameter.

See Also

[SalvageErasedFile \(page 307\)](#), [ScanErasedFiles \(page 309\)](#)

readdir

Obtains information about the next matching file using the attributes set by calling `SetReaddirAttribute`

Local Servers: blocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <dirent.h>
```

```
DIR *readdir (  
    DIR *dirP);
```

Parameters

dirP

(IN/OUT) Specifies the structure to receive information about the next matching file.

Return Values

Returns a pointer to an object of the DIR structure type containing information about the next matching file or directory.

If an error occurs, such as when there are no more matching file names, NULL is returned and `errno` and `NWerrno` are set. (Unless NULL is returned, ignore values in `errno` and `NWerrno`.)

Remarks

`readdir` can be called repeatedly to obtain the list of file and directory names contained in the directory specified by the path name given to the `opendir` function.

The `closedir` function must be called to close the directory and free the memory allocated by the `opendir` function.

The date and time fields are not in the DOS date/time format. It is easily put in the DOS format by swapping the high word with the low word.

Beginning with Release 9 of the NW SDK, `readdir` returns long names in the `d_name` field of the `dirent` structure if the target namespace is previously set to something other than DOS by calling `SetTargetNameSpace`. To have use of this long name functionality, you must compile with the `dirent.h` file included with Release 9 or later. In addition, with NetWare versions lower than 5, you might need CLIBAUx.NLM loaded on the server for symbol resolution. (Currently `readdir` support for spaces other than DOS is available only on calls to the local server.)

NOTE: The position in the structure of the `d_name` field prior to Release 9 has been assumed by the new `d_nameDOS` field to ensure backward compatibility, and the `d_name` field has been moved to the end of the structure. The new code puts the DOS name space name at the `d_nameDOS` field offset so old code will still work. This can all be done with relative ease because `CLIB` allocates the memory.

NOTE: To have `readdir` return all files for the pattern `*.*` for the long name space, call `UnAugmentAsterisk` before calling `opendir`. See [opendir \(page 296\)](#) or [UnAugmentAsterisk \(page 324\)](#) for details.

See [Using readdir\(\): Example \(NDK: Sample Code\)](#).

See Also

[closedir \(page 143\)](#), [opendir \(page 296\)](#), [SetReaddirAttribute \(page 316\)](#), [UnAugmentAsterisk \(page 324\)](#)

remove

Deletes a specified file

Local Servers: blocking

Remote Servers: blocking

Classification: ANSI

Platform: NLM

Service: File System

Syntax

```
#include <stdio.h>
#include <unistd.h>

int remove (
    const char *filename);
```

Parameters

filename

(IN) Specifies the string containing the full or relative path of the file to be deleted (maximum 255 characters, including the NULL terminator).

Return Values

Returns a value of 0 if successful, nonzero otherwise. When an error has occurred, `errno` contains a value indicating the type of error that has been detected.

Remarks

`remove` also works on the DOS partition.

`remove` causes a file to be marked for deletion. A file marked for deletion is not actually erased until the space it occupies is needed by another file. The current connection must have Delete rights to the file.

Wildcard specifiers are allowed for the `filename` parameter.

The `SalvageErasedFile` function can be used to salvage a file that has been marked for deletion but not yet purged.

The `SetCurrentNameSpace` function sets the name space which is used for parsing the path input to `remove`.

NOTE: For NetWare versions before 4.x, `remove` works only with the DOS name space for remote servers.

See Also

[PurgeErasedFile \(page 298\)](#), [SalvageErasedFile \(page 307\)](#), [unlink \(page 325\)](#)

rename

Renames a specified file

Local Servers: blocking

Remote Servers: blocking

Classification: ANSI

Platform: NLM

Service: File System

Syntax

```
#include <stdio.h>
#include <unistd.h>

int rename (
    const char *old,
    const char *new);
```

Parameters

old

(IN) Points to a string containing the full or relative path of the name of the file to be renamed (maximum 255 characters, including the NULL terminator).

new

(IN) Points to a string containing the full or relative path of the new file name to replace the old file name (maximum 255 characters, including the NULL terminator).

Return Values

Returns a value of 0 if successful, nonzero otherwise.

Remarks

NOTE: rename works only with the DOS and LONG name spaces. However, [NWSetNameSpaceEntryName \(page 549\)](#) can rename files in other name spaces.

Wildcard specifiers are allowed for the `old` and `new` parameters.

rename can also rename directories. However, if a wildcard is specified, only matching files (not directories) are renamed.

The current connection number must have Modify privileges. If a wildcard is specified, the current connection must also have See File rights. To move a file, the current connection must have Delete and Read rights for the file to be moved and Create rights in the destination.

To move a directory requires Delete rights to the directory to be moved and Create in the destination. The above-mentioned rights are also required for all directories and files in the subdirectory tree. Additionally, Create, See File, and Read rights are required to move deleted files; without these rights, deleted files are purged.

See Also

[FileServerFileCopy \(page 144\)](#), [NWGetNamespaceEntryName \(page 472\)](#),
[NWSetNamespaceEntryName \(page 549\)](#), [SetCurrentNamespace \(page 444\)](#),
[SetTargetNamespace \(page 446\)](#)

rmdir

Removes (deletes) the specified directory

Local Servers: blocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <unistd.h>

int rmdir (
    const char *pathname);
```

Parameters

pathname

(IN) Specifies either the absolute or relative directory path containing the directory to delete.

Return Values

Returns a value of 0 if successful, nonzero otherwise. If an error occurs, `errno` and `NetWareErrno` are set.

Remarks

`rmdir` also works on the DOS partition.

The directory must not contain any files or directories.

The `SetCurrentNameSpace` function sets the name space which is used for parsing the path input to `rmdir`.

NOTE: For NetWare versions before 4.x, `rmdir` works with only the DOS name space for remote servers.

See Also

[chdir \(page 140\)](#), [getcwd \(page 146\)](#), [mkdir \(page 151\)](#)

SalvageErasedFile

Salvages a file that has been marked for deletion

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfinfo.h>

int SalvageErasedFile (
    char *pathname,
    long  sequenceNumber,
    char *newFileName);
```

Parameters

pathname

(IN) Specifies the string containing the path name of the erased file to be salvaged (maximum 255 characters, including the NULL terminator).

sequenceNumber

(IN) Specifies which version of the specified file to restore.

newFileName

(IN) Points to a NULL-terminated string containing the name to give the erased file when it is restored (maximum 13 characters, including the NULL terminator).

See [Salvaging Files: Example](#) (*NDK: Sample Code*).

Return Values

0	0x00	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

A file marked for deletion with the `remove` or `unlink` function can be recovered by calling the `SalvageErasedFile` function.

The `pathname` parameter can be an absolute path with a volume name, or it can be relative to the current working directory.

The `sequenceNumber` parameter is obtained from the `ScanErasedFiles` function.

The `newFileName` parameter can be from 1 to 8 characters long and can also include an extension of from 1 to 3 characters. All letters must be uppercase and the string must be NULL-terminated.

The current connection must have Create rights in the specified directory.

The `SetCurrentNameSpace` function sets the name space that is used for parsing the path input to `rmdir`.

NOTE: `rmdir` currently works only in the DOS name space. However, you can salvage a file in other name spaces in the following way. Call the `SetCurrentNameSpace` function to change to the DOS name space. Then call the `ScanErasedFiles` function to get the DOS names of the files you want to salvage. The DOS names are returned in the structure that the `ScanErasedFiles` function uses. You can then salvage the files, supplying their DOS names to the `pathname` parameter. After you have salvaged the files, they still have directory entries in the other name spaces that are loaded just as they did before they were deleted.

See Also

[PurgeErasedFile \(page 298\)](#), [ScanErasedFiles \(page 309\)](#)

ScanErasedFiles

Returns information about deleted files

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfinfo.h>

int ScanErasedFiles (
    char *pathname,
    long *nextEntryNumber,
    DIR *deletedFileInfo);
```

Parameters

pathname

(IN) Specifies the string containing the path specification of the directory to view (maximum 255 characters, including the NULL terminator).

nextEntryNumber

(IN/OUT) Points to the entry number of the next file (-1 initially).

deletedFileInfo

(OUT) Points to the DIR structure.

Return Values

0	0x00	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

ScanErasedFiles can be called repeatedly to obtain the list of file names contained in the directory specified by the `pathname` parameter. Files marked for deletion can be scanned to obtain information about who deleted the files and when they were deleted.

The `pathname` parameter can be an absolute path with a volume name or it can be relative to the current working directory. Do not include a wildcard character at the end of the path. In the following example, the erased files in the DIR1 directory on the SYS volume are scanned:

SYS:DIR1

The current connection must have See File rights in the specified directory.

The SetCurrentNameSpace function sets the name space that is used for parsing the path input to ScanErasedFiles.

NOTE: ScanErasedFiles currently works only in the DOS name space. However, you can scan erased files for another name space. Call the SetCurrentNameSpace function to change to the DOS name space. Then call ScanErasedFiles, supplying a DOS path name.

ScanErasedFiles returns DOS names for the files that have been erased. You can then use those names to either salvage the files by calling the SalvageErasedFile function or purge them by calling the PurgeErasedFile function.

See Also

[PurgeErasedFile \(page 298\)](#), [SalvageErasedFile \(page 307\)](#)

SetExtendedFileAttributes

Sets the extended attributes byte for a file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwinfo.h>

int SetExtendedFileAttributes (
    char *filePath,
    BYTE  extendedFileAttributes);
```

Parameters

filePath

(IN) Specifies the string containing the relative or absolute (including the volume name) path specification of the file whose extended attributes are being changed (maximum 255 characters, including the NULL terminator).

extendedFileAttributes

(IN) Specifies the new extended attributes for the file.

Return Values

0	0x00	ESUCCESS
254	0xFE	ERR_INCORRECT_ACCESS_PRIVILEGES
255	0xFF	ERR_NO_FILES_FOUND

Remarks

SetExtendedFileAttributes sets the extended file attributes for a file by passing a file path and an extended file attributes byte. The current connection must have Modify rights to the file.

SetExtendedFileAttributes overwrites the first byte of the existing file attributes with the value in the `extendedFileAttributes` parameter. The byte definition follows:

- 3 Don't Suballocate (set this bit to disallow suballocation on this entry)
- 4 Transaction (used by TTS)
- 6 Read Audit (unused)

7 Write Audit (unused)

If the Transaction bit is set in the `extendedFileAttributes` parameter byte, TTS tracks all writes to the file during a transaction. A transaction file cannot be deleted or renamed until the transaction bit is turned off by calling `SetExtendedFileAttributes`.

NOTE: Do not confuse the first attributes byte with true extended attributes, which can be manipulated by calling the Extended Attribute functions.

The `SetCurrentNameSpace` function sets the name space which is used for parsing the path input to `SetExtendedFileAttributes`.

NOTE: For NetWare versions before 4.x, `SetExtendedFileAttributes` works only with the DOS name space for remote servers.

See Also

[GetExtendedFileAttributes \(page 147\)](#)

SetFileInfo

Sets file information for a file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfinfo.h>

int SetFileInfo (
    char *filePath,
    BYTE searchAttributes,
    LONG fileAttributes,
    char *creationDateAndTime,
    char *lastAccessDate,
    char *lastUpdateDateAndTime,
    char *lastArchiveDateAndTime,
    LONG fileOwnerID);
```

Parameters

filePath

(IN) Points to the string containing the path specification of the file to be changed (maximum 255 characters, including the NULL terminator).

searchAttributes

(IN) Specifies the type of the file for which to set file information.

fileAttributes

(IN) Specifies the file attributes to be assigned to the file.

creationDateAndTime

(IN) Points to the creation date and time to be assigned to the file (DOS format, 4 bytes).

lastAccessDate

(IN) Points to the last access date to be assigned to the file (DOS format, bytes 1 and 2).

lastUpdateDateAndTime

(IN) Points to the last update date and time to be assigned to the file (DOS format, 4 bytes).

lastArchiveDateAndTime

(IN) Points to the last archived date and time to be assigned to the file (DOS format, 4 bytes).

fileOwnerID

(IN) Specifies the unique object ID to be assigned as the new owner.

Return Values

0	0x00	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

SetFileInfo sets file information by passing the file path, the search attributes byte, and specific file information. File information includes file attributes, extended file attributes, creation date and time, last access date, last update date and time, file owner, and last archived date and time.

SetFileInfo expects the date and time to be in DOS format. The date and time field from readdir is not in the DOS date/time format but can be used by swapping the high word with the low word.

SetFileInfo requires that the requesting workstation have Supervisor rights to the file(s) being modified.

The `filePath` parameter can specify an absolute or a relative path. An absolute file path appears in the following format:

```
volume: directory1\...\directory\file name
```

A relative file path includes a file name and (optionally) one or more antecedent directory names.

A file name can be from 1 to 8 characters long and can include a 1- to 3-character extension. All letters must be upper case. The last item in the `filePath` parameter must be a valid file name specification. No wildcard specifiers are allowed.

The `searchAttributes` parameter can have the following values:

0x00	Normal files
0x02	Normal and hidden files
0x04	Normal and system files
0x06	Normal, hidden, and system files

SetFileInfo can assign file attributes to a specified file by passing a new value in the `fileAttributes` parameter. The following bits are defined for byte 0:

- 0 Read Only
- 1 Hidden
- 2 System
- 3 Execute Only
- 4 Subdirectory
- 5 Archive
- 6 Undefined

7 Share

The following bits are defined for byte 1, the extended attributes byte:

- 3 Don't Suballocate (set this bit to disallow suballocation on this entry)
- 4 Transaction (used by TTS)
- 6 Read Audit (unused)
- 7 Write Audit (unused)

In NetWare 3.0 and above, you can set four file attributes in byte 2, bits 0, 1, 2, and 4. In NetWare 4.x, 5.x, and 6.x, you can set bit 7:

- 0 Immediate Purge
- 1 Rename Inhibit
- 2 Delete Inhibit
- 3 Copy Inhibit
- 7 Data Migration Inhibit

NetWare 4.x, 5.x, and 6.x also allow you to set file attributes in an additional byte, byte 3:

- 0 Data Save Key (used for data migration)
- 1 Immediately Compress File (or all files in subdirectory)
- 2 Data Stream Compressed
- 3 Do Not Compress This Entry
- 4 Create a Hard link Entry (for NFS)
- 5 Cannot Compress Data Stream
- 6 Attribute Archive Bit

The `creationDateAndTime`, `lastUpdateDateAndTime`, and `lastArchiveDateAndTime` parameters occupy bytes 0, 1, 2, and 3.

The application can change the owner of the file by passing the object ID number of the new owner in the `fileOwnerID` parameter.

The `SetCurrentNameSpace` function sets the name space which is used for parsing the path input to `SetFileInfo`.

See Also

[NWSetDirEntryInfo \(page 277\)](#), [readdir \(page 300\)](#)

SetReaddirAttribute

Sets the attributes that are to be used when searching for files and directories by calling the readdir function

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfileio.h>

int SetReaddirAttribute (
    DIR *dirP,
    unsigned long newAttribute);
```

Parameters

dirP

(IN) Points to the DIR structure obtained by calling opendir or readdir.

newAttribute

(IN) Specifies the new attribute.

Return Values

Returns a value of 0 if successful, nonzero otherwise.

Remarks

SetReaddirAttribute can be called any time after the DIR structure has been obtained from the opendir function. The modified search attributes are in effect for calling the readdir function.

The following search attributes are defined:

<code>_A_NORMAL</code>	Normal file; read/write permitted
<code>_A_RDONLY</code>	Read-only file
<code>_A_HIDDEN</code>	Hidden file
<code>_A_SYSTEM</code>	System file
<code>_A_VOLID</code>	Volume ID entry
<code>_A_SUBDIR</code>	Subdirectory

See Also

[closedir \(page 143\)](#), [opendir \(page 296\)](#), [readdir \(page 300\)](#)

_splitpath

Splits a full path name into four components consisting of a server/volume name, directory path, file name, and file name extension

Local Servers: blocking

Remote Servers: N/A

Platform: NLM

Service: File System

Syntax

```
#include <nwfileio.h>
```

```
void _splitpath (  
    const char *path,  
    char       *drive,  
    char       *dir,  
    char       *fname,  
    char       *ext);
```

Parameters

path

(IN) Specifies the string containing the full path name to split.

drive

(OUT) Points to the server/volume name or drive letter. The maximum string length is 64.

dir

(OUT) Points to the directory path. The maximum string length is 254.

fname

(OUT) Points to the base name of the file without an extension. The maximum string length is 8.

ext

(OUT) Points to the file name extension, including the leading period. The maximum string length is 4.

Remarks

`_splitpath` returns the drive letter in the `drive` parameter. If you pass it a NetWare path, `_splitpath` returns the NetWare server/volume in the `drive` parameter.

This function is coded to work only with the DOS namespace (8.3).

The `drive`, `dir`, `fname`, and `ext` parameters are not filled in if they are NULL. For each component of the full path name that is not present, its corresponding buffer is set to an empty string.

See [Using `_makepath` and `_splitpath`: Example](#) (*NDK: Sample Code*).

See Also

[_makepath](#) (page 149)

stat

Retrieves the status of a specified file or directory

Local Servers: blocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <stat.h>

int stat (
    const char *path,
    struct stat *statblk);
```

Parameters

path

(IN) Points to a string containing the path of the directory or file for which status is to be obtained (maximum 255 characters, including the NULL terminator).

statblk

(OUT) Points to the [stat \(page 368\)](#) containing information about the file.

Return Values

Returns a value of 0 when the information is successfully obtained. Otherwise, a value of -1 is returned and errno is set to indicate the type of error that occurred.

Remarks

stat (Function) returns information in the stat (Structure) located at the address indicated by the statblk parameter.

The SYS\STAT.H header file contains definitions for the stat (Structure) and describes the contents of the fields.

The time and date in the stat (Structure) are in calendar format.

Beginning with Release 9 of the NW SDK, stat (Function) returns long names in the d_name field of the stat (Structure) if the st_name field is set to something other than DOS. You must compile with the stat.h file included with Release 9 or later and link with the new nwpre.obj and is valid only when calling stat (Function) on the local server.

The current connection must have See File rights.

The SetCurrentNameSpace function sets the name space which is used for parsing the path input to stat (Function).

NOTE: For NetWare versions before 4.x, stat (Function) works only with the DOS name space for remote servers.

See Also

[fstat](#) (Single and Intra-File Services)

tmpnam

Generates a unique string for use as a valid temporary file name

Local Servers: blocking

Remote Servers: blocking

Classification: ANSI

Platform: NLM

Service: File System

Syntax

```
#include <stdio.h>
#include <unistd.h>

char *tmpnam (
    char *buffer);
```

Parameters

buffer

(OUT) Points to the buffer to receive the generated temporary file name.

Return Values

If you pass a NULL pointer, tmpnam leaves the temporary file name in an internal static buffer and returns a pointer to that buffer.

Remarks

Be aware that the internal static buffer is modified every time tmpnam is called, whether or not you pass a NULL pointer. If you want to preserve the temporary file name currently stored in the internal static buffer, copy it to another buffer (by calling the strcpy function) before calling tmpnam again.

If you pass a pointer to your created array, tmpnam leaves the temporary file name in that array and returns a pointer to it. tmpnam simply returns the pointer you have supplied. It does no error checking to ensure that your array is big enough to accommodate the file name. The array should be at least L_tmpnam characters in length, where L_tmpnam is 13 characters (12 for the DOS 8.3 characters plus one for the NULL terminator).

See [Using tmpnam: Example \(NDK: Sample Code\)](#).

See Also

[access \(page 138\)](#)

umask

Sets the file permission mask (part of the thread group context)

Local Servers: blocking

Remote Servers: N/A

Platform: NLM

Service: File System

Syntax

```
#include <stat.h>

int umask (
    int  permission);
```

Parameters

permission

(IN) Specifies the file permission mask to be used to update the permission of the current process.

Return Values

Returns the previous value of the `permission` parameter.

Remarks

The file permission mask is used to modify the permission setting of new files created by the `creat`, `open`, or `sopen` function. If a bit in the mask is on, the corresponding bit in the requested permission value for the file is disallowed.

The `permission` parameter is a constant expression involving the constants `S_IWRITE` and `S_IREAD` as defined in `SYS\STAT.H`.

<code>S_IWRITE</code>	Write permission
<code>S_IREAD</code>	Read permission

See Also

[chmod \(page 141\)](#), [creat](#), [open](#), [sopen](#) (Single and Intra-File Services)

UnAugmentAsterisk

Makes the *.* pattern return all files and subdirectory names for the long (OS/2) name space

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfileio.h>

void UnAugmentAsterisk (
    int  yesno);
```

Parameters

yesno

(IN) Specifies whether to return all files and subdirectory names for the long name space:

TRUE	The *.* pattern returns all file and subdirectory names for the long name space.
FALSE	(default) The *.* pattern does not return file and directory names in the long name space that contain more than one dot.

Remarks

The default behavior for `opendir` and `readdir` is to interpret a pattern of *.* to return only those file and directory names that contain only one dot. Therefore, the pattern *.* guarantees that all files are returned for the DOS name space only. Calling `UnAugmentAsterisk` allows you to use *.* to return all file and directory names for the long name space as well.

NOTE: The name of the function refers to the fact that the high bit for the asterisk character in the pattern is set by default. This function reverses this setting.

See Also

[opendir \(page 296\)](#), [readdir \(page 300\)](#)

unlink

Deletes the specified file

Local Servers: blocking

Remote Servers: blocking

Classification: ANSI

Platform: NLM

Service: File System

Syntax

```
#include <unistd.h>

int unlink (
    const char *filename);
```

Parameters

filename

(IN) Points to a string containing the absolute or relative path of the file to delete (maximum 255 characters, including the NULL terminator).

Return Values

Returns a value of 0 if successful, nonzero otherwise. When an error has occurred, `errno` contains a value indicating the type of error that has been detected.

Remarks

`unlink` also works on the DOS partition.

A file marked for deletion is not actually erased by `unlink` until the space it occupies is needed by another file.

Wildcard specifiers are allowed for the `filename` parameter.

The `SalvageErasedFile` function can be called to salvage a file that has been marked for deletion but not yet purged.

The current connection must have Delete rights to the file.

See [Using unlink\(\): Example \(NDK: Sample Code\)](#).

See Also

[PurgeErasedFile \(page 298\)](#), [remove \(page 302\)](#), [SalvageErasedFile \(page 307\)](#)

UseAccurateCaseForPaths

Changes the case-specific manipulation behavior of file and path CLIB functions.

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File System

Syntax

```
#include <nwfileio.h>

void UseAccurateCaseForPaths (
    int  yesno);
```

Parameters

yesno

(IN) Specifies whether new NetWare file or directory names should be converted to uppercase characters:

FALSE (default value) Convert the specified file or directory name to uppercase characters.

TRUE Do not convert the specified file or directory name to uppercase characters.

Remarks

UseAccurateCaseForPaths is most useful in the LONG name space and has no effect on the DOS name space.

See Also

[SetCurrentNameSpace \(page 444\)](#)

utime

Updates the modification time for the specified file

Local Servers: blocking

Remote Servers: blocking

Classification: POSIX

Platform: NLM

Service: File System

Syntax

```
#include <utime.h>

int utime (
    const char          *filename,
    const struct utimbuf *times);
```

Parameters

filename

(IN) Points to a string containing the name of the file whose modification time is to be updated (maximum 255 characters, including the NULL terminator).

times

(IN) Points to the structure containing the modification time.

Return values

Returns a value of 0 when the time was successfully recorded. A value of -1 indicates an error occurred. If an error occurs, `errno` is set.

Remarks

If the `filename` parameter specifies a directory, the modification time and date are updated and the last accessed date is ignored (since directories do not have a last accessed date).

If the `times` parameter is NULL, the current time is used for the update. Otherwise, the `times` parameter must point to an object of the struct `utimbuf` type.

The modification time is taken from the `modtime` field in the `utimbuf` structure, and the last accessed date is taken from the `actime` field. (DOS has no notion of "accessed time." Therefore when time is being set on the DOS partition, the value in the `actime` field is undefined, and only the `modtime` field is of concern.)

The current connection must have Modify rights or Write rights to update the last modification time. It must also have Modify or Read rights to update the last accessed date.

The SetCurrentNameSpace function sets the name space which is used for parsing the path input to utime.

NOTE: For NetWare versions before 4.x, utime works only with the DOS name space for remote servers.

This documentation alphabetically lists the File System structures and describes their purpose, syntax, and fields.

- ◆ “CONN_USING_FILE” on page 330
- ◆ “CONNS_USING_FILE” on page 332
- ◆ “DIR” on page 334
- ◆ “DIR_SPACE_INFO” on page 337
- ◆ “ModifyStructure” on page 339
- ◆ “NW_EXT_FILE_INFO” on page 341
- ◆ “NW_FILE_INFO2” on page 345
- ◆ “NW_FILE_INFO2_EXT” on page 347
- ◆ “NW_LIMIT_LIST” on page 349
- ◆ “NWDIR_INFO” on page 351
- ◆ “NWENTRY_INFO” on page 353
- ◆ “NWET_INFO” on page 355
- ◆ “NWET_INFO_EXT” on page 356
- ◆ “NWFILE_INFO” on page 357
- ◆ “OPEN_FILE_CONN” on page 359
- ◆ “OPEN_FILE_CONN_CTRL” on page 362
- ◆ “SEARCH_DIR_INFO” on page 363
- ◆ “SEARCH_FILE_INFO” on page 366
- ◆ “stat” on page 368
- ◆ “TRUSTEE_INFO” on page 371
- ◆ “utimbuf” on page 372
- ◆ “VOLUME_STATS” on page 373
- ◆ “VOLUME_INFO” on page 375

CONN_USING_FILE

Defines file information for a file opened by a connection

Service: File System

Defined In: nwfile.h

Structure

```
typedef struct {
    NWCONN_NUM    connNumber ;
    nuint16       taskNumber ;
    nuint8        lockType ;
    nuint8        accessControl ;
    nuint8        lockFlag ;
} CONN_USING_FILE;
```

Delphi Structure

```
uses calwin32

CONN_USING_FILE = Record
    connNumber : NWCONN_NUM;
    taskNumber : nuint16;
    lockType : nuint8;
    accessControl : nuint8;
    lockFlag : nuint8
End;
```

Fields

connNumber

Specifies the logical connection number of a workstation using the file.

taskNumber

Specifies the number of the task which opened the file. A given connection may have several task numbers associated with the same file.

lockType

Specifies how the file is locked.

accessControl

Specifies how the file is accessed.

lockFlag

Specifies whether the file is locked.

Remarks

The `lockType` field can have the following values:

0x01 Locked
0x02 Open shareable
0x04 Logged
0x08 Open Normal
0x40 TTS holding
0x80 Transaction flag set

The `accessControl` field can have the following values:

0x01 Open for read by this client
0x02 Open for write by this client
0x04 Deny read requests from others
0x08 Deny write requests from others
0x10 File detached
0x20 TTS holding detach
0x40 TTS holding open

The `lockFlag` field can have the following values:

0x00 Not locked
0xFE Locked by a file lock
0xFF Locked by begin share file set

CONNS_USING_FILE

Returns a list of connections having a specified file open

Service: File System

Defined In: nwfile.h

Structure

```
typedef struct {
    nuint16      nextRequest ;
    nuint16      useCount ;
    nuint16      openCount ;
    nuint16      openForReadCount ;
    nuint16      openForWriteCount ;
    nuint16      denyReadCount ;
    nuint16      denyWriteCount ;
    nuint8       locked ;
    nuint8       forkCount ;
    nuint16      connCount ;
    CONN_USING_FILE connInfo [70];
} CONNS_USING_FILE;
```

Delphi Structure

```
uses calwin32

CONNS_USING_FILE = Record
    nextRequest : nuint16;
    useCount : nuint16;
    openCount : nuint16;
    openForReadCount : nuint16;
    openForWriteCount : nuint16;
    denyReadCount : nuint16;
    denyWriteCount : nuint16;
    locked : nuint8;
    forkCount : nuint8;
    connCount : nuint16;
    connInfo : Array[0..69] Of CONN_USING_FILE
End;
```

Fields

nextRequest

Specifies the sequence in subsequent calls to the NWScanConnectionsUsingFile function.

useCount

Specifies the number of tasks having the file opened or logged.

openCount

Specifies the number of tasks having opened or logged the file.

openForReadCount

Specifies the number of logical connections having the file open for reading.

openForWriteCount

Specifies the number of logical connections having the file open for writing.

denyReadCount

Specifies the number of logical connections having denied other connections access to the file.

denyWriteCount

Specifies the number of logical connections having denied other connections read access to the file.

locked

Specifies whether the file is locked exclusively (0=not locked exclusively).

forkCount

Specifies the number of forks associated with the file.

connCount

Specifies the number of connections using the file.

connInfo

Specifies an array of CONN_USING_FILE structures specifying how each connection is using the file.

DIR

Holds information about a directory entry

Service: File System

Defined In: dirent.h

Structure

```
typedef struct dirent {
    unsigned long    d_attr ;
    unsigned short   d_time ;
    unsigned short   d_date ;
    long             d_size ;
    ino_t             d_ino ;
    dev_t            d_dev ;
    unsigned long    d_cdatetime ;
    unsigned long    d_adatetime ;
    unsigned long    d_bdatetime ;
    long             d_uid ;
    unsigned long    d_archivedID ;
    unsigned long    d_updatedID ;
    char             d_nameDOS [13];
    unsigned short   d_inheritedRightsMask ;
    unsigned char    d_originatingNameSpace ;
    unsigned long    d_ddatetime ;
    unsigned long    d_deletedID ;
    char             d_name [255+1];
} DIR;
```

Fields

d_attr

Specifies the attribute as defined in NWFATTR.H.

d_time

Specifies the modification time in DOS format.

d_date

Specifies the modification date in DOS format.

d_size

Specifies the size (files only).

d_ino

Specifies the serial number.

d_dev

Specifies the volume number.

d_cdatetime

Specifies the creation date and time in DOS format.

d_adatetime

Specifies the last access date (files only) in DOS format.

d_bdatetime

Specifies the last archive date and time in DOS format.

d_uid

Specifies the owner ID (object ID).

d_archivedID

Specifies the object ID that last archived the file.

d_updateID

Specifies the object ID that last updated the file.

d_nameDOS

Specifies the DOS name space name.

d_inheritedRightsMask

Specifies the inherited rights mask.

d_originatingNameSpace

Specifies the creating name space.

d_ddatetime

Specifies the date and time the entry was deleted (used by the ScanErasedFiles function only).

d_deletedID

Specifies the object ID that deleted the file (used by the ScanErasedFiles function only).

d_name

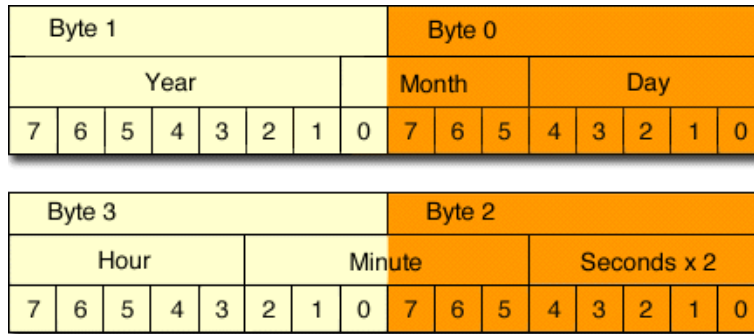
Specifies the name space name of the entry.

Remarks

The stack size might need to be increased (by using a link directive) when using the DIR structure especially in functions where the structure is used recursively such as: [opendir \(page 296\)](#), [readdir \(page 300\)](#), and [ScanErasedFiles \(page 309\)](#).

Date and time fields use standard DOS format as explained in the following graphic:

Figure 11-1 *Date and Time Fields*



The year bits contain the number of years elapsed since 1980. That value is added to 1980 and thus provides an accurate count to the year 2000 and beyond.

The seconds are calculated in 2-second intervals, so the value in the seconds bits must be multiplied by 2 to get the accurate number of seconds. For example a value of 15 means that 30 seconds have elapsed toward the next minute.

`d_time` uses bytes two and three.

`d_date` uses bytes 0 and 1.

`d_cdatetime`, `d_adatetime`, and `d_bdatetime` use all four bytes.

DIR_SPACE_INFO

Returns directory space information

Service: File System

Defined In: nwdirect.h

Structure

```
typedef struct {
    nuint32    totalBlocks ;
    nuint32    availableBlocks ;
    nuint32    purgeableBlocks ;
    nuint32    notYetPurgeableBlocks ;
    nuint32    totalDirEntries ;
    nuint32    availableDirEntries ;
    nuint32    reserved ;
    nuint8     sectorsPerBlock ;
    nuint8     volLen ;
    nuint8     volName [NW_MAX_VOLUME_NAME_LEN];
} DIR_SPACE_INFO;
```

Delphi Structure

```
uses calwin32

DIR_SPACE_INFO = packed Record
    totalBlocks : nuint32;
    availableBlocks : nuint32;
    purgeableBlocks : nuint32;
    notYetPurgeableBlocks : nuint32;
    totalDirEntries : nuint32;
    availableDirEntries : nuint32;
    reserved : nuint32;
    sectorsPerBlock : nuint8;
    volLen : nuint8;
    volName : Array[0..NW_MAX_VOLUME_NAME_LEN-1] Of nuint8
End;
```

Fields

totalBlocks

Specifies the total blocks in the volume.

availableBlocks

Specifies the number of available blocks.

purgeableBlocks

Specifies the number of recoverable blocks recovered by purging (0 if the NWGetDirSpaceInfo function is called with a directory handle of 0).

notYetPurgeableBlocks

Specifies the number of blocks not yet purgeable (0 if the NWGetDirSpaceInfo function is called with a directory handle of 0).

totalDirEntries

Specifies the number of entries in the directory.

availableDirEntries

Specifies the number of available entries remaining.

reserved

Is reserved for future use.

sectorsPerBlock

Specifies the number of sectors per block.

volLen

Specifies the length of the volName field.

volName

Specifies the name of the volume.

Remarks

All sizes are returned based on the block size of the volume (64 KB).

ModifyStructure

Holds information used in changing a directory entry

Service: File System

Defined In: nwdir.h

Structure

```
typedef struct {
    BYTE    *MModifyName ;
    LONG    MFileAttributes ;
    LONG    MFileAttributesMask ;
    WORD    MCreateDate ;
    WORD    MCreateTime ;
    LONG    MOwnerID ;
    WORD    MLastArchivedDate ;
    WORD    MLastArchivedTime ;
    LONG    MLastArchivedID ;
    WORD    MLastUpdatedDate ;
    WORD    MLastUpdatedTime ;
    LONG    MLastUpdatedID ;
    WORD    MLastAccessedDate ;
    WORD    MInheritanceGrantMask ;
    WORD    MInheritanceRevokeMask ;
    int     MMaximumSpace ;
    LONG    MLastUpdatedInSeconds ;
} ModifyStructure;
```

Fields

MModifyName

Points to the new directory name.

MFileAttributes

Specifies new file attributes.

MFileAttributesMask

Specifies new file attribute mask.

MCreateDate

Specifies new creation date.

MCreateTime

Specifies new creation time

MOwnerID

Specifies new owner ID.

MLastArchivedDate

Specifies the last archived date.

MLastArchivedTime

Specifies the last archived time.

MLastArchivedID

Specifies the last archived ID.

MLastUpdatedDate

Specifies the last updated date.

MLastUpdatedTime

Specifies the last updated time.

MLastUpdatedID

Specifies the last updated ID.

MLastAccessedDate

Specifies the last accessed date.

MInheritanceGrantMask

Specifies the inheritance grant mask.

MInheritanceRevokeMask

Specifies the inheritance revoke mask.

MMaximumSpace

Specifies the maximum space.

MLastUpdatedInSeconds

Specifies the last update in seconds.

NW_EXT_FILE_INFO

Returns extended file information

Service: File System

Defined In: nwdentry.h

Structure

```
typedef struct {
    nuint32    sequence ;
    nuint32    parent ;
    nuint32    attributes ;
    nuint8     uniqueID ;
    nuint8     flags ;
    nuint8     nameSpace ;
    nuint8     nameLength ;
    nuint8     name [12];
    nuint32    creationDateAndTime ;
    nuint32    ownerID ;
    nuint32    lastArchiveDateAndTime ;
    nuint32    lastArchiverID ;
    nuint32    updateDateAndTime ;
    nuint32    lastUpdatorID ;
    nuint32    dataForkSize ;
    nuint32    dataForkFirstFAT ;
    nuint32    nextTrusteeEntry ;
    nuint8     reserved [36];
    nuint16    inheritedRightsMask ;
    nuint16    lastAccessDate ;
    nuint32    deletedFileTime ;
    nuint32    deletedDateAndTime ;
    nuint32    deleterID ;
    nuint8     reserved2 [16];
    nuint32    otherForkSize [2];
} NW_EXT_FILE_INFO;
```

Delphi Structure

```
uses calwin32
```

```
NW_EXT_FILE_INFO = packed Record
    sequence : nuint32;
    parent : nuint32;
    attributes : nuint32;
    uniqueID : nuint8;
    flags : nuint8;
    nameSpace : nuint8;
    nameLength : nuint8;
    name : Array[0..11] Of nuint8;
    creationDateAndTime : nuint32;
    ownerID : nuint32;
```

```
lastArchiveDateAndTime : nuint32;
lastArchiverID : nuint32;
updateDateAndTime : nuint32;
lastUpdaterID : nuint32;
dataForkSize : nuint32;
dataForkFirstFAT : nuint32;
nextTrusteeEntry : nuint32;
reserved : Array[0..35] Of nuint8;
inheritedRightsMask : nuint16;
lastAccessDate : nuint16;
deletedFileTime : nuint32;
deletedDateAndTime : nuint32;
deletorID : nuint32;
reserved2 : Array[0..15] Of nuint8;
otherForkSize : Array[0..1] Of nuint32
End;
```

Fields

sequence

Specifies the sequence for iteratively scanning entries (-1 initially).

parent

Specifies the directory entry ID of parent directory.

attributes

Specifies the attributes of the entry.

uniqueID

Specifies the unique entry ID.

flags

Is reserved for future use.

nameSpace

Specifies the name space creating the entry.

nameLength

Specifies the maximum number of characters in the name.

name

Specifies the entry name.

creationDateAndTime

Specifies when the entry was created.

ownerID

Specifies the object ID of the owner.

lastArchiveDateAndTime

Specifies when the entry was last archived.

lastArchiverID

Specifies the ID of the object last archiving the entry.

updateDateAndTime

Specifies the date and time when the entry was last modified.

lastUpdaterID

Specifies the ID of the object that last modified the entry.

dataForkSize

Specifies the number of bytes in the file.

dataForkFirstFAT

Specifies the first file allocation table (FAT) entry for the indicated file.

nextTrusteeEntry

Specifies the next trustee of the entry.

reserved

Is reserved for future use.

inheritedRightsMask

Specifies the Inherited Rights Mask for the entry.

lastAccessDate

Specifies the date when the entry was last accessed.

deletedFileTime

Specifies the time when the file was deleted.

deletedDateAndTime

Specifies the date and time when the entry was deleted.

deletorID

Specifies the ID of the object deleting the entry.

reserved2

Is reserved for future use.

otherForkSize

Specifies a two-part array, which specifies the file size for the data stream supported by the given name space and the first FAT entry for the name space-specific data stream respectively.

Remarks

See [Section 20.2, “Attribute Values,” on page 593](#) for the possible values for the `attributes` field.

The `nameSpace` field can have the following values:

0 NW_NS_DOS

- 1 NW_NS_MAC
- 2 NW_NS_NFS
- 3 NW_NS_FTAM
- 4 NW_NS_LONG

The `inheritedRightsMask` field can have the following values:

C Value	Delphi Value	Value Description
0x0000	\$0000	TR_NONE
0x0001	\$0001	TR_READ
0x0002	\$0002	TR_WRITE
0x0004	\$0004	TR_OPEN
0x0004	\$0004	TR_DIRECTORY
0x0008	\$0008	TR_CREATE
0x0010	\$0010	TR_DELETE
0x0010	\$0010	TR_ERASE
0x0010	\$0020	TR_OWNERSHIP
0x0020	\$0020	TR_ACCESS_CTRL
0x0040	\$0040	TR_FILE_SCAN
0x0040	\$0040	TR_SEARCH
0x0040	\$0040	TR_FILE_ACCESS
0x0080	\$0080	TR_MODIFY
0x01FB	\$01FB	TR_ALL
0x0100	\$0100	TR_SUPERVISOR
0x00FB	\$00FB	TR_NORMAL

NW_FILE_INFO2

Holds file information

Service: File System

Defined In: nwfile.h

Structure

```
typedef struct {
    nuint8    fileAttributes ;
    nuint8    extendedFileAttributes ;
    nuint32   fileSize ;
    nuint16   creationDate ;
    nuint16   lastAccessDate ;
    nuint32   lastUpdateDateAndTime ;
    nuint32   fileOwnerID ;
    nuint32   lastArchiveDateAndTime ;
    nstr8     fileName [260];
} NW_FILE_INFO2;
```

Delphi Structure

```
uses calwin32

NW_FILE_INFO2 = packed Record
    fileAttributes : nuint8;
    extendedFileAttributes : nuint8;
    fileSize : nuint32;
    creationDate : nuint16;
    lastAccessDate : nuint16;
    lastUpdateDateAndTime : nuint32;
    fileOwnerID : nuint32;
    lastArchiveDateAndTime : nuint32;
    fileName : Array[0..259] Of nstr8
End;
```

Fields

fileAttributes

Specifies the file attributes (for values, see Remarks).

extendedFileAttributes

Specifies the file extended attributes (for values, see Remarks).

fileSize

Specifies the size of the file.

creationDate

Specifies when the file was created.

lastAccessDate

Specifies when the file was last accessed.

lastUpdateDateAndTime

Specifies when the file was last updated.

fileOwnerID

Specifies the object ID of the owner.

lastArchiveDateAndTime

Specifies when the file was last archived.

fileName

Specifies the name of the file (long names are supported).

Remarks

The `fileAttributes` field can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x01	\$01	FA_READ_ONLY
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x08	\$08	FA_EXECUTE_ONLY
0x10	\$10	FA_DIRECTORY
0x20	\$20	FA_NEEDS_ARCHIVED
0x80	\$80	FA_SHAREABLE

The `extendedFileAttributes` field can have the following values:

C Value	Delphi Value	Value Name
0x10	\$10	FA_TRANSACTIONAL
0x20	\$20	FA_INDEXED
0x40	\$40	FA_READ_AUDIT
0x80	\$80	FA_WRITE_AUDIT

NW_FILE_INFO2_EXT

Holds file information

Service: File System

Defined In: nwfile.h

Structure

```
typedef struct {
    nuint8    fileAttributes ;
    nuint8    extendedFileAttributes ;
    nuint32   fileSize ;
    nuint16   creationDate ;
    nuint16   lastAccessDate ;
    nuint32   lastUpdateDateAndTime ;
    nuint32   fileOwnerID ;
    nuint32   lastArchiveDateAndTime ;
    nstr8     fileName [766];
} NW_FILE_INFO2_EXT;
```

Fields

fileAttributes

Specifies the file attributes (for values, see Remarks).

extendedFileAttributes

Specifies the file extended attributes (for values, see Remarks).

fileSize

Specifies the size of the file.

creationDate

Specifies when the file was created.

lastAccessDate

Specifies when the file was last accessed.

lastUpdateDateAndTime

Specifies when the file was last updated.

fileOwnerID

Specifies the object ID of the owner.

lastArchiveDateAndTime

Specifies when the file was last archived.

fileName

Specifies the name of the file (long names are supported), using UTF-8 characters.

Remarks

The `fileAttributes` field can have the following values:

C Value	Value Name
0x00	FA_NORMAL
0x01	FA_READ_ONLY
0x02	FA_HIDDEN
0x04	FA_SYSTEM
0x08	FA_EXECUTE_ONLY
0x10	FA_DIRECTORY
0x20	FA_NEEDS_ARCHIVED
0x80	FA_SHAREABLE

The `extendedFileAttributes` field can have the following values:

C Value	Value Name
0x10	FA_TRANSACTIONAL
0x20	FA_INDEXED
0x40	FA_READ_AUDIT
0x80	FA_WRITE_AUDIT

NW_LIMIT_LIST

Returns disk space information about the restrictions along the directory path

Service: File System

Defined In: nwdirect.h

Structure

```
typedef struct {
    nuint8      numEntries ;
    struct {
        nuint8   level ;
        nuint32  max ;
        nuint32  current ;
    } list[102];
} NW_LIMIT_LIST
```

Delphi Structure

```
uses calwin32

NW_LIMIT_LIST = Packed Record
    numEntries :  nuint8 ;
    list : Array[0..101] of Record
        level :  nuint8 ;
        max :   nuint32 ;
        current :  nuint32 ;
    End;
End;
```

Fields

numEntries

Specifies the number of entries returned in the structure.

level

Specifies the distance from the directory to the root for each entry.

max

Specifies the maximum amount of space (in 4 KB sizes) assigned to a directory for each entry.

current

Specifies the amount of space (in 4 KB sizes) assigned to a directory minus the amount of space used by a directory and its subdirectories for each entry.

Remarks

`level` specifies to which directory `max` and `current` refer. The specified directory is always the first entry. For other entries, to find out what parent directory is being referred to, parse the directory path to match the level of each entry after the first one in the list. (The root of the volume is zero.)

If the `max` field for a directory is `0x7FFFFFFF`, there is no restriction for the entry. If the `max` field is greater than `0x7FFFFFFF`, the limit is zero. For all other values, `max` represents the restriction in 4K increments. You can multiply `max` by 4 to get the restrictions in KB. The same is true for the `current` field. The `max` and `current` fields are allowed to be negative so a valid space-in-use value may be calculated.

`current` is equal to `max` minus the space that is already in use by the directory and its subdirectories, which can be obtained by subtracting `current` from `max`. When `max` is set to a value greater than `0x7FFFFFFF`, the space in use is equal to zero minus `current`. (`current` will be negative so the answer will be positive.) Do not directly use `current` in this case because it might be a negative number.

The space-in-use value can be calculated by subtracting the value of the `current` field from the value of the `max` field.

NWDIR_INFO

Defines entry information for directories

Service: File System

Defined In: nwdentry.h

Structure

```
typedef struct {
    nuint32    lastModifyDateAndTime ;
    nuint32    nextTrusteeEntry ;
    nuint8     reserved [48];
    nuint32    maximumSpace ;
    nuint16    inheritedRightsMask ;
    nuint8     reserved2 [14];
    nuint32    volObjectID ;
    nuint8     reserved3 [8];
} NWDIR_INFO;
```

Delphi Structure

```
uses calwin32
```

```
NWDIR_INFO = packed Record
    lastModifyDateAndTime : nuint32;
    nextTrusteeEntry : nuint32;
    reserved : Array[0..47] Of nuint8;
    maximumSpace : nuint32;
    inheritedRightsMask : nuint16;
    reserved2 : Array[0..13] Of nuint8;
    volObjectID : nuint32;
    reserved3 : Array[0..7] Of nuint8
End;
```

Fields

lastModifyDateAndTime

Specifies when the directory was last updated.

nextTrusteeEntry

Specifies the next trustee entry in the subdirectory.

reserved

Is reserved for future use.

maximumSpace

Specifies the maximum space available in the subdirectory.

inheritedRightsMask

Specifies the Inherited Rights Mask.

reserved2

Is reserved for future use.

volObjectID

Specifies the volume object ID.

reserved3

Is reserved for future use.

Remarks

The `inheritedRightsMask` field can have the following values:

C Value	Delphi Value	Value Description
0x0000	\$0000	TR_NONE
0x0001	\$0001	TR_READ
0x0002	\$0002	TR_WRITE
0x0004	\$0004	TR_OPEN
0x0004	\$0004	TR_DIRECTORY
0x0008	\$0008	TR_CREATE
0x0010	\$0010	TR_DELETE
0x0010	\$0010	TR_ERASE
0x0020	\$0020	TR_OWNERSHIP
0x0020	\$0020	TR_ACCESS_CTRL
0x0040	\$0040	TR_FILE_SCAN
0x0040	\$0040	TR_SEARCH
0x0040	\$0040	TR_FILE_ACCESS
0x0080	\$0080	TR_MODIFY
0x01FB	\$01FB	TR_ALL
0x0100	\$0100	TR_SUPERVISOR
0x00FB	\$00FB	TR_NORMAL

NWENTRY_INFO

Defines directory entry information

Service: File System

Defined In: nwdentry.h

Structure

```
typedef struct {
    nuint32    sequence ;
    nuint32    parent ;
    nuint32    attributes ;
    nuint8     uniqueID ;
    nuint8     flags ;
    nuint8     nameSpace ;
    nuint8     nameLength ;
    nuint8     name [12];
    nuint32    creationDateAndTime ;
    nuint32    ownerID ;
    nuint32    lastArchiveDateAndTime ;
    nuint32    lastArchiverID ;
    union {
        NWFILE_INFO    file ;
        NWDIR_INFO     dir ;
    } info;
} NWENTRY_INFO;
```

Delphi Structure

```
uses calwin32

NWENTRY_INFO = packed Record
    sequence : nuint32;
    parent : nuint32;
    attributes : nuint32;
    uniqueID : nuint8;
    flags : nuint8;
    nameSpace : nuint8;
    nameLength : nuint8;
    name : Array[0..11] Of nuint8;
    creationDateAndTime : nuint32;
    ownerID : nuint32;
    lastArchiveDateAndTime : nuint32;
    lastArchiverID : nuint32;
    case Integer of
        1:(file1: NWFILE_INFO);
        2:(dir : NWDIR_INFO);
    end;
End;
```

Fields

sequence

Specifies the sequence for iteratively scanning entries (-1 initially).

parent

Specifies the directory handle to parent directory.

attributes

Specifies the entry attributes.

uniqueID

Specifies the unique entry ID.

flags

Is reserved.

nameSpace

Specifies the name space creating the entry.

nameLength

Specifies the length of the name field.

name

Specifies the entry name.

creationDateAndTime

Specifies when the entry was created.

ownerID

Specifies the object ID of the owner.

lastArchiveDateAndTime

Specifies when the entry was last archived.

lastArchiverID

Specifies the ID of the object last archiving the entry.

Remarks

See [Section 20.2, “Attribute Values,” on page 593](#) for the possible values for the `attributes` field.

The `nameSpace` field can have the following values:

- 0 NW_NS_DOS
- 1 NW_NS_MAC
- 2 NW_NS_NFS
- 3 NW_NS_FTAM
- 4 NW_NS_LONG

NWET_INFO

Returns directory entry trustee information

Service: File System

Defined In: nwdentry.h

Structure

```
typedef struct {
    nstr8          entryName [16];
    nuint32       creationDateAndTime ;
    nuint32       ownerID ;
    nuint32       sequenceNumber ;
    TRUSTEE_INFO trusteeList [20];
} NWET_INFO;
```

Delphi Structure

```
uses calwin32

NWET_INFO = packed Record
    entryName : Array[0..15] Of nstr8;
    creationDateAndTime : nuint32;
    ownerID : nuint32;
    sequenceNumber : nuint32;
    trusteeList : Array[0..19] Of TRUSTEE_INFO
End;
```

Fields

entryName

Set to zero.

creationDateAndTime

Set to zero.

ownerID

Set to zero.

sequenceNumber

Specifies the sequence for iteratively scanning entries.

trusteeList

Specifies an array of up to 20 TRUSTEE_INFO structures.

NWET_INFO_EXT

Returns directory entry trustee information

Service: File System

Defined In: nwdentry.h

Structure

```
typedef struct {
    nstr8          entryName [16];
    nuint32       creationDateAndTime ;
    nuint32       ownerID ;
    nuint32       sequenceNumber ;
    TRUSTEE_INFO trusteeList [100];
} NWET_INFO_EXT;
```

Fields

entryName

Set to zero.

creationDateAndTime

Set to zero.

ownerID

Set to zero.

sequenceNumber

Specifies the sequence for iteratively scanning entries.

trusteeList

Specifies an array of up to 100 TRUSTEE_INFO structures.

NWFILE_INFO

Defines entry information for files

Service: File System

Defined In: nwdentry.h

Structure

```
typedef struct {
    nuint32    updateDateAndTime ;
    nuint32    updatorID ;
    nuint32    fileSize ;
    nuint8     reserved [44];
    nuint16    inheritedRightsMask ;
    nuint16    lastAccessDate ;
    nuint8     reserved2 [28];
} NWFILE_INFO;
```

Delphi Structure

```
uses calwin32

NWFILE_INFO = packed Record
    updateDateAndTime : nuint32;
    updatorID : nuint32;
    fileSize : nuint32;
    reserved : Array[0..43] Of nuint8;
    inheritedRightsMask : nuint16;
    lastAccessDate : nuint16;
    reserved2 : Array[0..27] Of nuint8
End;
```

Fields

updateDateAndTime

Specifies when the file was last updated.

updatorID

Specifies the ID of the object that last updated the file.

fileSize

Specifies the size of the file.

reserved

Is reserved for future use.

inheritedRightsMask

Specifies the Inherited Rights Mask for the file.

lastAccessDate

Specifies when the file was last accessed

reserved2

Is reserved for future use.

Remarks

The `inheritedRightsMask` field can have the following values:

C Value	Delphi Value	Value Description
0x0000	\$0000	TR_NONE
0x0001	\$0001	TR_READ
0x0002	\$0002	TR_WRITE
0x0004	\$0004	TR_OPEN
0x0004	\$0004	TR_DIRECTORY
0x0008	\$0008	TR_CREATE
0x0010	\$0010	TR_DELETE
0x0010	\$0010	TR_ERASE
0x0020	\$0020	TR_OWNERSHIP
0x0020	\$0020	TR_ACCESS_CTRL
0x0040	\$0040	TR_FILE_SCAN
0x0040	\$0040	TR_SEARCH
0x0040	\$0040	TR_FILE_ACCESS
0x0080	\$0080	TR_MODIFY
0x01FB	\$01FB	TR_ALL
0x0100	\$0100	TR_SUPERVISOR
0x00FB	\$00FB	TR_NORMAL

OPEN_FILE_CONN

Returns information about the open files for a connection

Service: File System

Defined In: nwfile.h

Structure

```
typedef struct {
    nuint16    taskNumber ;
    nuint8     lockType ;
    nuint8     accessControl ;
    nuint8     lockFlag ;
    nuint8     volNumber ;
    nuint32    parent ;
    nuint32    dirEntry ;
    nuint8     forkCount ;
    nuint8     nameSpace ;
    nuint8     nameLen ;
    nstr8      fileName [255];
} OPEN_FILE_CONN;
```

Delphi Structure

```
uses calwin32

OPEN_FILE_CONN = packed Record
    taskNumber : nuint16;
    lockType : nuint8;
    accessControl : nuint8;
    lockFlag : nuint8;
    volNumber : nuint8;
    parent : nuint32;
    dirEntry : nuint32;
    forkCount : nuint8;
    nameSpace : nuint8;
    nameLen : nuint8;
    fileName : Array[0..254] Of nstr8
End;
```

Fields

taskNumber

Specifies the number of the task which has this file opened (each file can have multiple task numbers).

lockType

Specifies how the file is locked.

accessControl

Specifies how the file is being accessed.

lockFlag

Specifies whether the file is locked.

volNumber

Specifies the volume number (SYS is always 0).

parent

Specifies the ID number for the parent directory.

dirEntry

Specifies the directory entry number.

forkCount

Specifies the number of forks associated with the file.

nameSpace

Specifies the name space creating the file.

nameLen

Specifies the number of bytes in the filename.

fileName

Specifies the name of file (long names are supported).

Remarks

The first four fields contain information similar to their counterparts in the [CONN_USING_FILE \(page 330\)](#) structure. The remaining fields identify the file and its name space.

The `lockType` field can have the following values:

- 0x01 Locked
- 0x02 Open shareable
- 0x04 Logged
- 0x08 Open Normal
- 0x40 TTS holding
- 0x80 Transaction flag set

The `accessControl` field can have the following values:

- 0x01 Open for read by this client
- 0x02 Open for write by this client
- 0x04 Deny read requests from others
- 0x08 Deny write requests from others
- 0x10 File detached
- 0x20 TTS holding detach
- 0x40 TTS holding open

The `lockFlag` field can have the following values:

0x00 Not locked

0xFE Locked by a file lock

0xFF Locked by begin share file set

The `nameSpace` field can have the following values:

0 `NW_NS_DOS`

1 `NW_NS_MAC`

2 `NW_NS_NFS`

3 `NW_NS_FTAM`

4 `NW_NS_LONG`

OPEN_FILE_CONN_CTRL

Returns a list of files a specified connection has open

Service: File System

Defined In: nwfile.h

Structure

```
typedef struct {
    nuint16  nextRequest ;
    nuint16  openCount ;
    nuint8   buffer [512];
    nuint16  curRecord ;
} OPEN_FILE_CONN_CTRL;
```

Delphi Structure

```
uses calwin32

OPEN_FILE_CONN_CTRL = packed Record
    nextRequest : nuint16;
    openCount : nuint16;
    buffer : Array[0..511] Of nuint8;
    curRecord : nuint16
End;
```

Fields

nextRequest

Specifies an iterator.

openCount

Specifies the number of OPEN_FILE_CONN structures contained in the `buffer` field.

buffer

Specifies the returned OPEN_FILE_CONN structure.

curRecord

Specifies the offset in the `buffer` field of the next record to return and is used internally by the `NWScanOpenFilesByConn2` function to track the next record to return in the OPEN_FILE_CONN structure.

SEARCH_DIR_INFO

Service: File System

Defined In: nwfile.h

Structure

```
typedef struct {
    nuint16    sequenceNumber ;
    nuint16    reserved1 ;
    nstr8      directoryName [15];
    nuint8     directoryAttributes ;
    nuint8     directoryAccessRights ;
    nuint16    createDate ;
    nuint16    createTime ;
    nuint32    owningObjectID :
    nuint16    reserved2 ;
    nuint16    directoryStamp ;
} SEARCH_DIR_INFO;
```

Delphi Structure

```
uses calwin32

SEARCH_DIR_INFO = packed Record
    sequenceNumber : nuint16;
    reserved1 : nuint16;
    directoryName : Array[0..14] Of nstr8;
    directoryAttributes : nuint8;
    directoryAccessRights : nuint8;
    padd1 : nuint8;
    createDate : nuint16;
    createTime : nuint16;
    padd2 : nuint16;
    owningObjectID : nuint32;
    reserved2 : nuint16;
    directoryStamp : nuint16
End;
```

Fields

sequenceNumber

Is reserved for future use.

reserved1

Is reserved for future use.

directoryName

Specifies the short name of the directory.

directoryAttributes

Specifies the attributes for the directory.

directoryAccessRights

Specifies the access rights.

createDate

Specifies the time the directory was created.

createTime

Specifies the date the directory was created.

owningObjectID

Specifies the ID of the object owning the directory.

reserved2

Is reserved for future use.

directoryStamp

Specifies 0xD1D1 when returned.

Remarks

The `directoryAttributes` field can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x10	\$10	FA_DIRECTORY

FA_DIRECTORY will always be in the bit mask for a directory.

The `directoryAccessRights` field can have the following values:

C Value	Delphi Value	Value Name
0x00	\$00	TA_NONE
0x01	\$01	TA_READ
0x02	\$02	TA_WRITE
0x04	\$04	TA_OPEN Obsolete in 3.x and above.
0x08	\$08	TA_CREATE

C Value	Delphi Value	Value Name
0x10	\$10	TA_DELETE
0x20	\$20	TA_OWNERSHIP
0x40	\$40	TA_SEARCH
0x80	\$80	TA_MODIFY
0xFB	\$FB	TA_ALL

SEARCH_FILE_INFO

Service: File System

Defined In: nwfile.h

Structure

```
typedef struct {
    nuint16    sequenceNumber ;
    nuint16    reserved ;
    nstr8      fileName [15];
    nuint8     fileAttributes ;
    nuint8     fileMode ;
    nuint32    fileLength ;
    nuint16    createDate ;
    nuint16    accessDate ;
    nuint16    updateDate ;
    nuint16    updateTime ;
} SEARCH_FILE_INFO;
```

Delphi Structure

```
uses calwin32

SEARCH_FILE_INFO = packed Record
    sequenceNumber : nuint16;
    reserved : nuint16;
    fileName : Array[0..14] Of nstr8;
    fileAttributes : nuint8;
    fileMode : nuint8;
    fileLength : nuint32;
    createDate : nuint16;
    accessDate : nuint16;
    updateDate : nuint16;
    updateTime : nuint16
End;
```

Fields

sequenceNumber

Is reserved.

reserved

Is reserved for future use.

fileName

Specifies the short name of the file.

fileAttributes

Specifies the attributes for the file.

fileMode

Specifies the access rights.

fileLength

Specifies the size of the file in bytes.

createDate

Specifies the date when the file was created.

accessDate

Specifies the date when the file was last accessed.

updateDate

Specifies the date when the file was last modified.

updateTime

Specifies the time when the file was last modified.

Remarks

The `fileAttributes` field can have the following values (may be ORed):

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x01	\$01	FA_READ_ONLY
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x08	\$08	FA_EXECUTE_ONLY
0x10	\$10	FA_DIRECTORY
0x20	\$20	FA_NEEDS_ARCHIVED
0x80	\$80	FA_SHAREABLE

The `fileMode` field can have the following values:

0x01 Open for read by this client

0x02 Open for write by this client

0x04 Deny read requests from others

0x08 Deny write requests from others

0x10 File detached

0x20 TTS holding detach

0x40 TTS holding open

stat

Holds information about the status of a file or directory

Service: File System

Defined In: sys\stat.h

Structure

```
struct stat {
    dev_t          st_dev ;
    ino_t          st_ino ;
    unsigned short st_mode ;
    short         st_nlink ;
    unsigned long  st_uid ;
    short         st_gid ;
    dev_t         st_rdev ;
    off_t         st_size ;
    time_t        st_atime ;
    time_t        st_mtime ;
    time_t        st_ctime ;
    time_t        st_btime ;
    unsigned long  st_attr ;
    unsigned long  st_archivedID ;
    unsigned long  st_updatedID ;
    unsigned short st_inheritedRightsMask ;
    unsigned char  st_originatingNameSpace ;
    unsigned char  st_name [255+1];
    size_t         st_blksize ;
    size_t         st_blocks ;
    unsigned int   st_flags ;
    unsigned long  st_spare [4];
};
```

Fields

st_dev

Specifies the volume number.

st_ino

Specifies the directory entry of the st_name.

st_mode

Specifies the emulated file mode.

st_nlink

Specifies the count of hard links (always 1).

st_uid

Specifies the object ID of the owner.

st_gid

Specifies the group ID (always 0).

st_rdev

Specifies the device type (always 0).

st_size

Specifies the total file size (files only).

st_atime

Specifies the last access date/time (files only) in calendar time (seconds since the Jan.1, 1970 (UTC)).

st_mtime

Specifies the last modify date/time and time in calendar time.

st_ctime

Specifies the date/time in calendar time that the file or directory was created.

st_btime

Specifies the time in calendar time since the entry was last archived.

st_attr

Specifies the file attribute as defined in NWFATTR.H.

st_archivedID

Specifies the ID of the user/object that last archived the entry.

st_updatedID

Specifies the ID of the user/object that last updated the entry.

st_inheritedRightsMask

Specifies the NDS inherited rights mask.

st_originatingNameSpace

Specifies the name space in which the file or directory was created (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

st_name

Specifies the name of the file or directory according to the set target name space (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

st_blksize

Specifies the block size for allocation (files only).

st_blocks

Specifies the count of blocks allocated to the file.

st_flags

Specifies user-defined flags.

st_spare

Reserved for future use.

TRUSTEE_INFO

Contains a directory trustee with the object rights

Service: File System

Defined In: nwdirect.h

Structure

```
typedef struct {
    nuint32  objectID ;
    nuint16  objectRights ;
} TRUSTEE_INFO;
```

Delphi Structure

```
uses calwin32

TRUSTEE_INFO = packed Record
    objectID : nuint32;
    objectRights : nuint16;
    reserved : nuint16;
End;
```

Fields

objectID

Specifies the ID of the object.

objectRights

Specifies the rights the object has on a directory.

utimbuf

Contains when the file was last accessed and modified

Service: File System

Defined In: utime.h

Structure

```
struct {  
    time_t    actime ;  
    time_t    modtime ;  
};
```

Fields

actime

Specifies the last time the file was accessed.

modtime

Specifies the last time the file was modified.

VOLUME_STATS

Holds volume information

Service: File System

Defined In: nwdir.h

Structure

```
typedef struct tagVOLUME_STATS {
    long    systemElapsedTime ;
    BYTE    volumeNumber ;
    BYTE    logicalDriveNumber ;
    WORD    sectorsPerBlock ;
    long    startingBlock ;
    WORD    totalBlocks ;
    WORD    availableBlocks ;
    WORD    totalDirectorySlots ;
    WORD    availableDirectorySlots ;
    WORD    maxDirectorySlotsUsed ;
    BYTE    isHashing ;
    BYTE    isRemovable ;
    BYTE    isMounted ;
    char    volumeName [17];
    LONG    purgeableBlocks ;
    LONG    notyetPurgeableBlocks ;
} VOLUME_STATS;
```

Fields

systemElapsedTime

Specifies the time in seconds since the system was brought up.

volumeNumber

Specifies the volume number (same as the Volume Table number for the server).

logicalDriveNumber

Specifies the logical drive number.

sectorsPerBlock

Specifies the number of 512-byte sectors in a block for the volume.

startingBlock

Specifies the starting block of the volume.

totalBlocks

Specifies the total number of blocks in the volume.

availableBlocks

Specifies the number of available blocks on the volume.

totalDirectorySlots

Specifies the total number of directory slots on the volume.

availableDirectorySlots

Specifies the number of available directory slots on the volume.

maxDirectorySlotsUsed

Specifies the maximum number of directory slots used on the volume.

isHashing

Specifies whether the volume is hashing.

isRemovable

Specifies whether the volume is removable (always non-zero for NetWare 3.x and 4.x):

non-zero Volume can be removed

0x00 Volume cannot be removed

isMounted

Specifies whether the volume is mounted.

volumeName

Specifies the volume name (2-15 characters plus the NULL terminator).

purgableBlocks

Specifies the number of purgeable blocks

notYetPurgableBlocks

Specifies the number of blocks not yet purgeable.

Remarks

The `volumeName` field cannot contain spaces or the following characters:

*	Asterisk
?	Question mark
:	Colon
/	Slash
\	Backslash

VOLUME_INFO

Contains volume information

Service: File System

Defined In: nwdir.h

Structure

```
typedef struct tagVOLUME_INFO {
    long    systemElapsedTime ;
    BYTE    volumeNumber ;
    BYTE    logicalDriveNumber ;
    WORD    sectorsPerBlock ;
    short   startingBlock ;
    LONG    totalBlocks ;
    LONG    availableBlocks ;
    LONG    totalDirectorySlots ;
    LONG    availableDirectorySlots ;
    BYTE    isHashing ;
    BYTE    isRemovable ;
    BYTE    isMounted ;
    char    volumeName [17];
    LONG    purgableBlocks ;
    LONG    notyetPurgableBlocks ;
} VOLUME_INFO;
```

Fields

systemElapsedTime

Specifies the time in seconds since the system was brought up.

volumeNumber

Specifies the volume number (same as the Volume Table number).

logicalDriveNumber

Specifies the logical drive number.

sectorsPerBlock

Specifies the number of 512-byte sectors in a block for the volume.

startingBlock

Specifies the starting block of the volume.

totalBlocks

Specifies the total number of blocks in the volume.

availableBlocks

Specifies the number of available blocks on the volume.

totalDirectorySlots

Specifies the total number of directory slots on the volume.

availableDirectorySlots

Specifies the number of available directory slots on the volume.

isHashing

Specifies whether the volume is hashing.

isRemovable

Specifies whether the volume is removable (always non-zero for NetWare 3.x and 4.x):

non-zero Volume can be removed

0x00 Volume cannot be removed

isMounted

Specifies whether the volume is mounted.

volumeName

Specifies the volume name (2-15 characters plus the NULL terminator).

purgableBlocks

Specifies the number of purgeable blocks

notYetPurgableBlocks

Specifies the number of blocks not yet purgeable.

Remarks

The `volumeName` field cannot contain spaces or the following characters:

*	Asterisk
?	Question mark
:	Colon
/	Slash
\	Backslash

This documentation describes File System Monitoring, its functions, and features.

File System Monitoring allows your NLM application to "hook" the file system functions that correspond to the list below. Before any of these functions that your NLM has registered for callback are executed by the NetWare OS, your NLM has the option of changing it, failing it, or simply making a record of its execution.

File System Monitoring allows you to:

- ◆ erase, open, create, rename, move, and close files
- ◆ create and delete directories
- ◆ modify directory entries
- ◆ rename name space entries
- ◆ salvage, purge, open, create, rename, and erase generic entities
- ◆ modify generic DOS information and generic name space information

12.1 Registering for Callback

The NetWare® OS transfers control to your NLM whenever it receives a request from any of its clients for a function that you have registered for monitoring.

Control is transferred to a function in your NLM that has restrictions imposed on it by the NetWare OS. This "callback function" is required to have parameters that the OS is expecting and can fill out. Your NLM or a system administrator can then use the information passed to the callback function by the OS to decide what action to take, if any, before or after the request is filled.

It's as if, when you call [NWAddFSMonitorHook \(page 386\)](#), your NLM is given a window through which the NetWare OS looks at every request for a file system function that you have registered for monitoring. Your NLM can then test each one against a selected set of conditions, such as the presence of a virus. In the event your NLM detects something suspicious, it can alter or fail the request or make a record of it for the system administrator to act upon later.

12.2 File Monitoring

What your monitoring function returns depends on whether it is a pre-execution callback or a post-execution callback:

- ◆ [“Pre-Execution and Post-Execution Monitoring” on page 378](#)
- ◆ [“Pre-Execution Callbacks” on page 379](#)
- ◆ [“Post-Execution Callbacks” on page 379](#)
- ◆ [“Callback Structures” on page 379](#)

12.2.1 Pre-Execution and Post-Execution Monitoring

When registering a callback function, you specify in the `callBackNumber` parameter whether the callback is made before or after the OS executes the function. Possible values for the `callBackNumber` include both a "pre" and "post" version for every OS function that can be monitored. The "pre" versions callback to your function before the OS function executes, whereas the "post" versions callback to your function after the OS function executes. If the callback occurs before the OS executes the function, your NLM can fail that function. Call [NWAddFSMonitorHook \(page 386\)](#) once for each function you want to be monitored.

The name space entry changing hooks and all generic hooks are used for monitoring functions called from other than DOS clients. These non-DOS hooks are supported only on NetWare® versions 3.12 and higher, while the remaining hooks are also supported on version 3.11. The following table lists the values for `callBackNumber` for each OS function:

Table 12-1 *Callback Functions for Monitoring File Operations*

OS Function to Monitor	Callback before OS Execution	Callback after OS Execution
file erasing	FSHOOK_PRE_ERASEFILE	FSHOOK_POST_ERASEFILE
file opening	FSHOOK_PRE_OPENFILE	FSHOOK_POST_OPENFILE
file creating	FSHOOK_PRE_CREATEFILE	FSHOOK_POST_CREATEFILE
file creating/ opening	FSHOOK_PRE_CREATE_OPENFILE	FSHOOK_POST_CREATE_OPENFILE
file renaming/ moving	FSHOOK_PRE_RENAME_OR_MOVE	FSHOOK_POST_RENAME_OR_MOVE
file closing	FSHOOK_PRE_CLOSEFILE	FSHOOK_POST_CLOSEFILE
directory creating	FSHOOK_PRE_CREATEDIR	FSHOOK_POST_CREATEDIR
directory deleting	FSHOOK_PRE_DELETEDIR	FSHOOK_POST_DELETEDIR
directory entry modification	FSHOOK_PRE_MODIFY_DIRENTRY	FSHOOK_POST_MODIFY_DIRENTRY
salvaging	FSHOOK_PRE_SALVAGE_DELETED	FSHOOK_POST_SALVAGE_DELETED
purging	FSHOOK_PRE_PURGE_DELETED	FSHOOK_POST_PURGE_DELETED
name space entry renaming	FSHOOK_PRE_RENAME_NS_ENTRY	FSHOOK_POST_RENAME_NS_ENTRY
generic salvaging	FSHOOK_PRE_GEN_SALVAGE_DELETED	FSHOOK_POST_GEN_SALVAGE_DELETED
generic purging	FSHOOK_PRE_GEN_PURGE_DELETED	FSHOOK_POST_GEN_PURGE_DELETED
generic opening/ creating	FSHOOK_PRE_GEN_OPEN_CREATE	FSHOOK_POST_GEN_OPEN_CREATE
generic renaming	FSHOOK_PRE_GEN_RENAME	FSHOOK_POST_GEN_RENAME
generic file erasing	FSHOOK_PRE_GEN_ERASEFILE	FSHOOK_POST_GEN_ERASEFILE

OS Function to Monitor	Callback before OS Execution	Callback after OS Execution
generic DOS information modification	FSHOOK_PRE_GEN_MODIFY_DOS_INFO	FSHOOK_POST_GEN_MODIFY_DOS_INFO
generic name space information modification	FSHOOK_PRE_GEN_MODIFY_NS_INFO	FSHOOK_POST_GEN_MODIFY_NS_INFO

12.2.2 Pre-Execution Callbacks

If you are registering a pre-execution function, it should return one parameter, a pointer to the structure returned for the OS function you are monitoring.

In the case of pre-execution callbacks, you have the option of failing the OS function and returning an error. If your NLM decides to fail a request, it should return one of the OS standard error codes (see NITERROR.H).

12.2.3 Post-Execution Callbacks

If you are registering a post-execution function, it should return 2 parameters, a pointer to the structure returned for the OS function and a completion code indicating whether or not the OS function completed successfully.

NOTE: The post-execution callback function must not sleep, because the fields in the return structure are subject to change.

12.2.4 Callback Structures

The following table summarizes the structures returned by file system monitoring callbacks:

FSHOOK_PRE_ERASEFILE	EraseFileCallBackStruct (page 400)
FSHOOK_POST_ERASEFILE	
FSHOOK_PRE_OPENFILE	OpenFileCallBackStruct (page 418)
FSHOOK_POST_OPENFILE	
FSHOOK_PRE_CREATEFILE	CreateFileCallBackStruct (page 395)
FSHOOK_POST_CREATEFILE	
FSHOOK_PRE_CREATE_OPENFILE	CreateAndOpenCallBackStruct (page 397)
FSHOOK_POST_CREATE_OPENFILE	
FSHOOK_PRE_RENAME_OR_MOVE	RenameMoveEntryCallBackStruct (page 422)
FSHOOK_POST_RENAME_OR_MOVE	
FSHOOK_PRE_CLOSEFILE	CloseFileCallBackStruct (page 392)
FSHOOK_POST_CLOSEFILE	

FSHOOK_PRE_CREATEDIR	CreateDirCallBackStruct (page 393)
FSHOOK_POST_CREATEDIR	
FSHOOK_PRE_DELETEDIR	DeleteDirCallBackStruct (page 399)
FSHOOK_POST_DELETEDIR	
FSHOOK_PRE_MODIFY_DIRENTRY	ModifyDirEntryCallBackStruct (page 415)
FSHOOK_POST_MODIFY_DIRENTRY	
FSHOOK_PRE_SALVAGE_DELETED	SalvageDeletedCallBackStruct (page 426)
FSHOOK_POST_SALVAGE_DELETED	
FSHOOK_PRE_PURGE_DELETED	PurgeDeletedCallBackStruct (page 421)
FSHOOK_POST_PURGE_DELETED	
FSHOOK_PRE_RENAME_NS_ENTRY	RenameNSEntryCallBackStruct (page 424)
FSHOOK_POST_RENAME_NS_ENTRY	
FSHOOK_PRE_GEN_SALVAGE_DELETED	GenericSalvageDeletedCBStruct (page 414)
FSHOOK_POST_GEN_SALVAGE_DELETED	
FSHOOK_PRE_GEN_PURGE_DELETED	GenericPurgeDeletedCBStruct (page 411)
FSHOOK_POST_GEN_PURGE_DELETED	
FSHOOK_PRE_GEN_OPEN_CREATE	GenericOpenCreateCBStruct (page 408)
FSHOOK_POST_GEN_OPEN_CREATE	
FSHOOK_PRE_GEN_RENAME	GenericRenameCBStruct (page 412)
FSHOOK_POST_GEN_RENAME	
FSHOOK_PRE_GEN_ERASEFILE	GenericEraseFileCBStruct (page 402)
FSHOOK_POST_GEN_ERASEFILE	
FSHOOK_PRE_GEN_MODIFY_DOS_INFO	GenericModifyDOSInfoCBStruct (page 404)
FSHOOK_POST_GEN_MODIFY_DOS_INFO	
FSHOOK_PRE_GEN_MODIFY_NS_INFO	GenericModifyNSInfoCBStruct (page 406)
FSHOOK_POST_GEN_MODIFY_NS_INFO	

12.3 Potential Uses

Novell® originally created File System Monitoring to fill a demand for a virus detection/protection hook. Because viruses can infect mission-critical files, this is a vitally important use of the service, but not the only one. File System Monitoring could also be used for any other network service that relies on monitoring file system requests. A couple of these are hot backup and version control.

12.3.1 Hot Backup

A hot backup NLM could register functions that create and modify files, putting the results in a special log file. Then, from time to time, it could back up all the new material to a specified medium. This would eliminate the need for humanly-executed backup.

12.3.2 Version Control

A version control NLM could keep a record of .obj files that have been created or modified and store a copy of the last one, along with all pertinent information, in a specified place.

12.4 File System Monitoring Functions

These are the two functions associated with File System Monitoring:

<code>NWAddFSMonitorHook</code>	Begin monitoring the file system
<code>NWRemoveFSMonitorHook</code>	Stop monitoring the file system

This documentation describes common tasks associated with File System Monitoring.

13.1 Writing a File System Monitor NLM

The four steps below are the essential parts of writing a file system monitor NLM. They are taken from the example NLM, FSHOOK.C, in the EXAMPLES directory.

1 Create your callback functions.

```
int openFileCallBackFunc (OpenFileCallBackStruct *ofcbs)
    static int cnt = 0;
    char    user[48];
    int     ccode;
    WORD    objType;
    long    objID;
    BYTE    loginTime[7];
    LONG    pc;
    BYTE    ps[255];
    BYTE    volName[16];
    LONG    prevThreadGroupID;
    prevThreadGroupID = SetThreadGroupID (mainThreadGroupID);
    ccode = GetConnectionInformation (ofcbs->connection, user,
                                     &objType, &objID, loginTime);

    if (ccode != 0)
        return 0xFF;
    printf("%dth OPEN request. by %s (connNum %d), ", ++cnt, user,
          ofcbs->connection);
    FEMapVolumeNumberToName (ofcbs->volume, volName);
    for (pc = 1; pc <= volName[0]; pc++)
        putchar (volName [pc]);
    putchar (':');
    FEMapVolumeAndDirectoryToPath (ofcbs->volume,
                                    ofcbs->dirBase, ps, &pc);

    if (ps[0])
        printNetWareStr (pc, ps);
    printNetWareStr (ofcbs->pathComponentCount, ofcbs->pathString);
    putchar ('\n');
    SetThreadGroupID (prevThreadGroupID);
    return 0;
}
```

Registering your callback functions tells the OS to transfer control to your NLM whenever a specified file system event is triggered. These callback functions can be thought of as "windows" to the file system, which are opened by calling [NWAddFSMonitorHook \(page 386\)](#).

`openFileCallBackFunc` receives the `OpenFileCallBackStruct` pointer and prints out some of its field values for informational purposes.

2 To begin monitoring, call [NWAddFSMonitorHook \(page 386\)](#).

Register your callback functions and start monitoring. `openFileCallbackFunc` has been registered to be called back after the OS executes the file opening function (by specifying `FSHOOK_POST_OPENFILE`).

```
ccode = NWAddFSMonitorHook(FSHOOK_PRE_OPENFILE,
    openFileCallbackFunc, &preOpenFileHandle);
if (ccode != 0)
{
    printf("nwaddfsmonitorhook error.  ccode: %#x, hook:
        openFile\n", ccode);
}
```

3 Wait for callbacks from the OS.

```
while (1)
    ThreadSwitchWithDelay(1000);    //sleep forever...until unloaded
```

Provide a mechanism, like sleeping forever (above), for keeping the NLM inactive but loaded and ready to respond to a callback from the OS.

4 Stop monitoring.

```
void ExitandRemoveMonitorHooks ()
{
    NWRemoveFSMonitorHook(FSHOOK_PRE_OPENFILE,
        openFileCallbackFunc);
}
```

Deregister the callback by calling [NWRemoveFSMonitorHook \(page 389\)](#).

File System Monitoring Functions

14

This documentation alphabetically lists the File System Monitoring functions and describes their purpose, syntax, parameters, and return values.

- ◆ [“NWAddFSMonitorHook” on page 386](#)
- ◆ [“NWRemoveFSMonitorHook” on page 389](#)

NWAddFSMonitorHook

Allows the application to monitor ("hook") various OS file system routines

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File System Monitoring

Syntax

```
#include <nwfshook.h>

LONG NWAddFSMonitorHook (
    LONG     callBackNumber,
    void     *callBackFunc,
    LONG     *callBackHandle);
```

Parameters

callBackNumber

(IN) Specifies which type of OS file system routine you want to hook.

callBackFunc

(IN) Points to the function that you want the OS to call (pass control to) when the hooked file system routine is going to be or has been called by a client or any NLM application on the local server.

callBackHandle

(OUT) Points to a handle that identifies the file system monitor hook. This handle is passed to NWRemoveFSMonitorHook when removing the hook.

Return Values

If NWAddFSMonitorHook succeeds, it returns 0 if the OS routine corresponding to `callBackNumber` was successfully "hooked." Otherwise, it returns errors.

Remarks

The `callBackNumber` parameter specifies the OS file system routine that you want to hook, and whether the `callBackFunc` is called before (a "pre OS call hook") or after (a "post OS call hook") the OS routine executes. The last eight sets of hooks (`FSHOOK_PRE/POST_RENAME_NS_ENTRY` and all generics) are used for tracking routines called from other than DOS clients (note that non-DOS hooks are available for use only with NetWare versions 3.12 and higher.) Hooks cannot be ORed, so you must call NWAddFSMonitorHook once for each routine

you want to be monitored. Values for `callbackNumber` and the OS routines that they hook are defined in `nwfshook.h` and listed below:

Table 14-1 *File System Hooks*

Functionality	Constant	Valid Versions
file erasing	FSHOOK_PRE_ERASEFILE	3.11 and higher
	FSHOOK_POST_ERASEFILE	
file opening	FSHOOK_PRE_OPENFILE	3.11 and higher
	FSHOOK_POST_OPENFILE	
file creating	FSHOOK_PRE_CREATEFILE	3.11 and higher
	FSHOOK_POST_CREATEFILE	
file creating/opening	FSHOOK_PRE_CREATE_OPENFILE	3.11 and higher
	FSHOOK_POST_CREATE_OPENFILE	
file renaming/moving	FSHOOK_PRE_RENAME_OR_MOVE	3.11 and higher
	FSHOOK_POST_RENAME_OR_MOVE	
file closing	FSHOOK_PRE_CLOSEFILE	3.11 and higher
	FSHOOK_POST_CLOSEFILE	
directory creating	FSHOOK_PRE_CREATEDIR	3.11 and higher
	FSHOOK_POST_CREATEDIR	
directory deleting	FSHOOK_PRE_DELETEDIR	3.11 and higher
	FSHOOK_POST_DELETEDIR	
directory entry modification	FSHOOK_PRE_MODIFY_DIRENTRY	3.11 and higher
	FSHOOK_POST_MODIFY_DIRENTRY	
salvaging	FSHOOK_PRE_SALVAGE_DELETED	3.11 and higher
	FSHOOK_POST_SALVAGE_DELETED	
purging	FSHOOK_PRE_PURGE_DELETED	3.11 and higher
	FSHOOK_POST_PURGE_DELETED	
name space entry renaming	FSHOOK_PRE_RENAME_NS_ENTRY	3.12, 4.x, 5.x, 6.x
	FSHOOK_POST_RENAME_NS_ENTRY	
generic salvaging	FSHOOK_PRE_GEN_SALVAGE_DELETED	3.12, 4.x, 5.x, 6.x
	FSHOOK_POST_GEN_SALVAGE_DELETED	
generic purging	FSHOOK_PRE_GEN_PURGE_DELETED	3.12, 4.x, 5.x, 6.x
	FSHOOK_POST_GEN_PURGE_DELETED	

Functionality	Constant	Valid Versions
generic opening/creating	FSHOOK_PRE_GEN_OPEN_CREATE	3.12, 4.x, 5.x, 6.x
	FSHOOK_POST_GEN_OPEN_CREATE	
generic renaming	FSHOOK_PRE_GEN_RENAME	3.12, 4.x, 5.x, 6.x
	FSHOOK_POST_GEN_RENAME	
generic file erasing	FSHOOK_PRE_GEN_ERASEFILE	3.12, 4.x, 5.x, 6.x
	FSHOOK_POST_GEN_ERASEFILE	
generic DOS information modification	FSHOOK_PRE_GEN_MODIFY_DOS_INFO	3.12, 4.x, 5.x, 6.x
	FSHOOK_POST_GEN_MODIFY_DOS_INFO	
generic name space information modification	FSHOOK_PRE_GEN_MODIFY_NS_INFO	3.12, 4.x, 5.x, 6.x
	FSHOOK_POST_GEN_MODIFY_NS_INFO	

The `callBackFunc` parameter points to the callback function you have created. The number of parameters you should declare in `callBackFunc` varies depending on when the OS calls back the function.

The first parameter for both types of callback function is a pointer to the structure returned by the OS for the OS file system routine that is being monitored (for example, if you are monitoring file opens, the OS would return an `OpenFileCallBackStruct`). These callback structures are defined in `nwfshook.h`.

If you have specified a pre OS call back hook, this is the only parameter for the `callBackFunc`. If you have specified a post OS call back hook, the `callBackFunc` receives a second parameter, a pointer to a LONG value which is the completion code of the OS routine that you have hooked. The following illustrates what these functions would look like if you are monitoring file opens:

```
int PreCallBackFunc (OpenFileCallBackStruct const *structure);
void PostCallBackFunc (OpenFileCallBackStruct const *structure, LONG
ccode);
```

Definitions of the structures returned by the callback function are described in [“File System Monitoring Structures” on page 391](#).

NOTE: If you specify a post OS call back hook, your callback function must not go to sleep, because the values in the callback structure can change before your thread wakes up again.

See Also

[NWRemoveFSMonitorHook \(page 389\)](#)

NWRemoveFSMonitorHook

Removes a "hook" that is monitoring an OS file system routine

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: File System Monitoring

Syntax

```
#include <nwfshook.h>
```

```
LONG NWRemoveFSMonitorHook (  
    LONG    callBackNumber,  
    LONG    callBackHandle);
```

Parameters

callBackNumber

(IN) Specifies the OS file system routine that you want to remove a hook from. See [NWAddFSMonitorHook \(page 386\)](#) for possible values for this parameter.

callBackHandle

(IN) Specifies the handle that was returned when the hook was added by calling [NWAddFSMonitorHook](#).

Return Values

If [NWRemoveFSMonitorHook](#) succeeds, it returns 0 if the hook corresponding to `callBackNumber` was successfully removed. Otherwise, it returns errors.

See Also

[NWAddFSMonitorHook \(page 386\)](#)

This documentation alphabetically lists the File System Monitoring structures and describes their purpose, syntax, and fields.

- ◆ “CloseFileCallBackStruct” on page 392
- ◆ “CreateDirCallBackStruct” on page 393
- ◆ “CreateFileCallBackStruct” on page 395
- ◆ “CreateAndOpenCallBackStruct” on page 397
- ◆ “DeleteDirCallBackStruct” on page 399
- ◆ “EraseFileCallBackStruct” on page 400
- ◆ “GenericEraseFileCBStruct” on page 402
- ◆ “GenericModifyDOSInfoCBStruct” on page 404
- ◆ “GenericModifyNSInfoCBStruct” on page 406
- ◆ “GenericOpenCreateCBStruct” on page 408
- ◆ “GenericPurgeDeletedCBStruct” on page 411
- ◆ “GenericRenameCBStruct” on page 412
- ◆ “GenericSalvageDeletedCBStruct” on page 414
- ◆ “ModifyDirEntryCallBackStruct” on page 415
- ◆ “OpenFileCallBackStruct” on page 418
- ◆ “PurgeDeletedCallBackStruct” on page 421
- ◆ “RenameMoveEntryCallBackStruct” on page 422
- ◆ “RenameNSEntryCallBackStruct” on page 424
- ◆ “SalvageDeletedCallBackStruct” on page 426

CloseFileCallbackStruct

Contains information about a close file operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {  
    LONG    connection ;  
    LONG    task ;  
    LONG    fileHandle ;  
} CloseFileCallbackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

fileHandle

Contains the NetWare file handle of the file.

CreateDirCallbackStruct

Contains information about a create directory operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    volume ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    pathComponentCount ;
    LONG    nameSpace ;
    LONG    directoryAccessMask ;
} CreateDirCallbackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

directoryAccessMask

Contains a bit mask by which the directory is to be accessed subsequently. This is the same bit mask used by `ModifyInheritedRightsMask`, as follows:

- 0 Read (file reads allowed)
- 1 Write (file writes allowed)
- 2 Reserved
- 3 Create (files can be created)
- 4 Delete (files can be deleted)
- 5 Access control (trustee rights can be assigned)
- 6 See files (files can be seen in directory scans)
- 7 Modify (files can be modified)
- 8 Supervisor (all rights are granted)

CreateFileCallbackStruct

Contains information about a create file operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    volume ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    pathComponentCount ;
    LONG    nameSpace ;
    LONG    createAttributeBits ;
    LONG    createFlagBits ;
    LONG    dataStreamNumber ;
} CreateFileCallbackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG

5 NT

createAttributeBits

Contains the file attributes that the file is to have when it is created.

createFlagBits

Contains flags that can be set to allow more flexibility in the create operation. These bits are listed in the following table.

DELETE_FILE_ON_CREATE_BIT	If the file already exists, it is deleted. This allows the file to be created again.
NO_RIGHTS_CHECK_ON_OPEN_BIT	The user's rights to the file are not checked when the file is opened.
NO_RIGHTS_CHECK_ON_CREATE_BIT	The user's rights to the file are not checked when the file is created.
FILE_WRITE_THROUGH_BIT	When a file write is performed, the write function does not return until the data is actually written to the disk.
ENABLE_IO_ON_COMPRESSED_DATA_BIT	Any subsequent I/O on this entry is compressed
LEAVE_FILE_COMPRESSED_DATA_BIT	After all I/O has been done, leave this file compressed

dataStreamNumber

Contains a number identifying the data stream type of the file or directory:

- 0 Primary Data Stream (DOS)
- 1 Macintosh Resource Fork
- 2 FRAM Extra Data Fork

CreateAndOpenCallBackStruct

Contains information about a create/open operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    volume ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    pathComponentCount ;
    LONG    nameSpace ;
    LONG    createAttributeBits ;
    LONG    requestedAccessRights ;
    LONG    createFlagBits ;
    LONG    dataStreamNumber ;
} CreateAndOpenCallBackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM

4 LONG

5 NT

createAttributeBits

Contains the file attributes that the file is to have when it is created.

requestedAccessRights

Indicates how the entry is to be opened, such as Read Only, Read Write, Compatibility mode, and so on. The bits in this mask are defined in the following figure.

- 0 Read only mode
- 1 Write only mode
- 2 Deny read mode
- 3 Deny write mode
- 4 Compatibility mode
- 6 File write through mode
- 8 Enable I/O on compressed data (NetWare 4.x)
- 9 Leave this file compressed (NetWare 4.x)
- 12 Always read ahead
- 13 Never read ahead

createFlagBits

Contains flags that can be set to allow more flexibility in the create operation. These bits are listed in the following table.

DELETE_FILE_ON_CREATE_BIT	If the file already exists, it is deleted. This allows the file to be created again.
NO_RIGHTS_CHECK_ON_OPEN_BIT	The user's rights to the file are not checked when the file is opened.
NO_RIGHTS_CHECK_ON_CREATE_BIT	The user's rights to the file are not checked when the file is created.
FILE_WRITE_THROUGH_BIT	When a file write is performed, the write function does not return until the data is actually written to the disk.
ENABLE_IO_ON_COMPRESSED_DATA_BIT	Any subsequent I/O on this entry is compressed
LEAVE_FILE_COMPRESSED_DATA_BIT	After all I/O has been done, leave this file compressed

dataStreamNumber

Contains a number identifying the data stream type of the file or directory:

- 0 Primary Data Stream (DOS)
- 1 Macintosh Resource Fork
- 2 FRAM Extra Data Fork

DeleteDirCallbackStruct

Contains information about a delete directory operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    volume ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    pathComponentCount ;
    LONG    nameSpace ;
} DeleteDirCallbackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

EraseFileCallbackStruct

Contains information about an erase file operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    volume ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    pathComponentCount ;
    LONG    nameSpace ;
    LONG    attributeMatchBits ;
} EraseFileCallbackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

0 DOS

1 MACINTOSH

2 NFS

3 FTAM

4 LONG

5 NT

attributeMatchBits

Contains a bit mask of the file attributes that are affected by this operation. That is, entries that have file attributes matching this bit mask are affected. For more about the file attributes mask, see [“File Attributes” on page 127](#). The bits of the first byte of the file attributes mask is as follows:

- 0 Read Only
- 1 Hidden
- 2 System
- 3 Execute Only
- 4 Subdirectory
- 5 Archive
- 6 Undefined
- 7 Share

GenericEraseFileCBStruct

Contains information about a generic erase file operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    volume ;
    LONG    pathComponentCount ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    nameSpace ;
    LONG    searchAttributes ;
} GenericEraseFileCBStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

0 DOS

1 MACINTOSH

2 NFS

3 FTAM

4 LONG

5 NT

searchAttributes

Contains a bit mask of the file attributes that are affected by this operation. That is, entries that have file attributes matching this bit mask are affected.

GenericModifyDOSInfoCBStruct

Contains information about a generic modify DOS information operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    volume ;
    LONG    pathComponentCount ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    nameSpace ;
    LONG    searchAttributes ;
    LONG    modifyMask ;
    void    *modifyInfo ;
} GenericModifyDOSInfoCBStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

0 DOS

1 MACINTOSH

2 NFS

3 FTAM

4 LONG

5 NT

searchAttributes

Contains field contains a bit mask of the file attributes that are affected by this operation. That is, entries that have file attributes matching this bit mask are affected.

modifyMask

Contains a bit mask that defines the items to be modified by this operation:

- 0 Name
- 1 Attributes
- 2 Creation Date
- 3 Creation Time
- 4 Creator ID
- 5 Archive Date
- 6 Archive Time
- 7 Archive ID
- 8 Modify Date
- 9 Modify Time
- 10 Modify ID
- 11 Last Access
- 12 Restrict (inheritance rights)
- 13 Maximum Space Allowed
- 14 Last Modified (in seconds)

modifyInfo

Contains the data that is to replace the old data for this entry.

GenericModifyNSInfoCBStruct

Contains information about a generic modify name space information operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    dataLength ;
    LONG    srcNameSpace ;
    LONG    dstNameSpace ;
    LONG    volume ;
    LONG    dirBase ;
    LONG    modifyMask ;
    void    *modifyInfo ;
} GenericModifyNSInfoCBStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

dataLength

Contains the size of the data in the `modifyInfo` field.

srcNameSpace

Contains the name space of the source:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

dstNameSpace

Contains the name space of the destination (see above).

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

modifyMask

Contains a bit mask that defines the items to be modified by this operation (see the following figure). Note that this bit mask differs slightly from the modify mask for the generic modify DOS information structure, in that it does not contain the "Last modified" bit:

- 0 Name
- 1 Attributes
- 2 Creation Date
- 3 Creation Time
- 4 Creator ID
- 5 Archive Date
- 6 Archive Time
- 7 Archive ID
- 8 Modify Date
- 9 Modify Time
- 10 Modify ID
- 11 Last Access
- 12 Restrict (inheritance rights)
- 13 Maximum Space Allowed

modifyInfo

Contains the data that is to replace the old data for this entry.

GenericOpenCreateCBStruct

Contains information about a generic open/create operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    volume ;
    LONG    pathComponentCount ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    nameSpace ;
    LONG    dataStreamNumber ;
    LONG    openCreateFlags ;
    LONG    searchAttributes ;
    LONG    createAttributes ;
    LONG    requestedAccessRights ;
    LONG    returnInfoMask ;
    LONG    *fileHandle ;
    BYTE    *openCreateAction ;
} GenericOpenCreateCBStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

pathComponentCount

Contains the number of components in the path.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

nameSpace

Contains the name space of the file or directory.

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

dataStreamNumber

Contains a number identifying the data stream type of the file or directory:

- 0 Primary Data Stream (DOS)
- 1 Macintosh Resource Fork
- 2 FRAM Extra Data Fork

openCreateFlags

Contains the operation requested, such as opening a file, creating a file, etc.:

- 0x01 Open
- 0x02 Truncate
- 0x08 Create

searchAttributes

Contains a bit mask of the file attributes that are affected by this operation. That is, entries that have file attributes matching this bit mask are affected.

createAttributes

Contains the attributes that are to be set when the entry is created.

requestedAccessRights

Indicates how the entry is to be opened, such as Read Only, Read Write, Compatibility mode, and so on. The bits in this mask are defined as follows:

- 0 Read only mode
- 1 Write only mode
- 2 Deny read mode
- 3 Deny write mode
- 4 Compatibility mode
- 6 File write through mode
- 8 Enable I/O on compressed data (NetWare 4.x)
- 9 Leave this file compressed (NetWare 4.x)
- 12 Always read ahead
- 13 Never read ahead

returnInfoMask

Contains a bit mask defining the information that is requested for this operation. This bit mask is defined as follows:

- 0 Entry name
- 1 Entry size

- 2 File attributes
- 3 Data stream information
- 4 Total data stream size
- 5 Extended attributes (EA) information
- 6 Archive information
- 7 Modify information
- 8 Creation information
- 9 Name space information
- 10 Directory information
- 11 Rights
- 12 Data stream size in sectors
- 13 Data stream logical size

fileHandle

Contains the NetWare file handle of the entry to be created.

openCreateAction

Contains the results of the requested action:

- 0x01 Open
- 0x02 Created
- 0x04 Truncated
- 0x08 Compressed (NetWare 4.0 only)
- 0xFF Bad Action

GenericPurgeDeletedCBStruct

Contains information about a generic purge deleted operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {  
    LONG    connection ;  
    LONG    nameSpace ;  
    LONG    sequence ;  
    LONG    volume ;  
    LONG    dirBase ;  
} GenericPurgeDeletedCBStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

nameSpace

Contains the name space of the file or directory:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

sequence

Contains the NetWare-internal number that was generated while scanning for deleted files.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

GenericRenameCBStruct

Contains information about a generic rename operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    nameSpace ;
    LONG    renameFlag ;
    LONG    searchAttributes ;
    LONG    srcVolume ;
    LONG    srcPathComponentCount ;
    LONG    srcDirBase ;
    BYTE    *srcPathString ;
    LONG    dstVolume ;
    LONG    dstPathComponentCount ;
    LONG    dstDirBase ;
    BYTE    *dstPathString ;
} GenericRenameCBStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

nameSpace

Contains the name space of the file or directory:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

renameFlag

Contains values defining rename options:

- 0x01 Allow renames to same name
- 0x02 Rename incompatibility mode
- 0x04 Only change names for the specified name space

searchAttributes

Contains field contains a bit mask of the file attributes that are affected by this operation. That is, entries that have file attributes matching this bit mask are affected.

srcVolume

Contains the volume number of the entry to be renamed.

srcPathComponentCount

Contains the number of path components for the source path.

srcDirBase

Contains the source directory base.

srcPathString

Contains the path string of the source.

dstVolume

Contains the volume number of the renamed entry.

dstPathComponentCount

Contains the number of path components for the destination path.

dstDirBase

Contains the destination directory base.

dstPathString

Contains the path string of the destination.

GenericSalvageDeletedCBStruct

Contains information about a generic salvage deleted operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    nameSpace ;
    LONG    sequence ;
    LONG    volume ;
    LONG    dirBase ;
    BYTE    *newName ;
} GenericSalvageDeletedCBStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

nameSpace

Contains the name space of the file or directory:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

sequence

Contains the NetWare-internal number that was generated while scanning for deleted files.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

newName

Contains the new name of the file or directory.

ModifyDirEntryCallbackStruct

Contains information about a modify directory operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG                connection ;
    LONG                task ;
    LONG                volume ;
    LONG                dirBase ;
    BYTE                *pathString ;
    LONG                pathComponentCount ;
    LONG                nameSpace ;
    LONG                attributeMatchBits ;
    LONG                targetNameSpace ;
    struct ModifyStructure *modifyVector ;
    LONG                modifyBits ;
    LONG                allowWildCardsFlag ;
} ModifyDirEntryCallbackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

0 DOS

1 MACINTOSH

2 NFS

- 3 FTAM
- 4 LONG
- 5 NT

attributeMatchBits

Contains a bit mask of the file attributes that are affected by this operation. That is, entries that have file attributes matching this bit mask are affected. For more about the file attributes mask, see [“File Attributes” on page 127](#). The bits of the first byte of the file attributes mask is defined as follows:

- 0 Read Only
- 1 Hidden
- 2 System
- 3 Execute Only
- 4 Subdirectory
- 5 Archive
- 6 Undefined
- 7 Share

targetNameSpace

Contains the name space of the entry that is to be changed (see the values for `nameSpace`, above).

modifyVector

Contains the modify vector used in the operation. See the discussion of [ModifyStructure \(page 339\)](#).

modifyBits

Contains the modify bits used in the operation:

- 0x0001L MModifyNameBit
- 0x0002L MFileAttributesBit
- 0x0004L MCreateDateBit
- 0x0008L MCreateTimeBit
- 0x0010L MOwnerIDBit
- 0x0020L MLastArchivedDateBit
- 0x0040L MLastArchivedTimeBit
- 0x0080L MLastArchivedIDBit
- 0x0100L MLastUpdatedDateBit
- 0x0200L MLastUpdatedTimeBit
- 0x0400L MLastUpdatedIDBit
- 0x0800L MLastAccessedDateBit
- 0x1000L MInheritanceRestrictionMaskBit
- 0x2000L MMaximumSpaceBit
- 0x4000L MLastUpdatedInSecondsBit

allowWildcardsFlag

Indicates whether wildcards are allowed in the path name:

Nonzero = Wildcards allowed
0 = No wildcards allowed.

See Also

[NWSetDirEntryInfo \(page 277\)](#)

OpenFileCallbackStruct

Contains information about an open file operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection;
    LONG    task;
    LONG    volume;
    LONG    dirBase;
    BYTE    *pathString;
    LONG    pathComponentCount;
    LONG    nameSpace;
    LONG    attributeMatchBits;
    LONG    requestedAccessRights;
    LONG    dataStreamNumber;
    LONG    *fileHandle;
} OpenFileCallbackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory. The value in this field is valid for the callback routine. Once the routine is called, the value is no longer valid. If you need this information outside of your callback routine, you need to copy and save the information.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

0 DOS

1 MACINTOSH

- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

attributeMatchBits

Contains a bit mask of the file attributes that are affected by this operation. That is, entries that have file attributes matching this bit mask are affected. For more about the file attributes mask, see [“File Attributes” on page 127](#). The bits of the first byte are defined as follows:

- 0 Read Only
- 1 Hidden
- 2 System
- 3 Execute Only
- 4 Subdirectory
- 5 Archive
- 6 Undefined
- 7 Share

requestedAccessRights

Indicates how the entry is to be opened, such as Read Only, Read Write, Compatibility mode, and so on. The bits in this mask are defined as follows:

- 0 Read only mode
- 1 Write only mode
- 2 Deny read mode
- 3 Deny write mode
- 4 Compatibility mode
- 6 File write through mode
- 8 Enable I/O on compressed data (NetWare 4.x)
- 9 Leave this file compressed (NetWare 4.x)
- 12 Always read ahead
- 13 Never read ahead

dataStreamNumber

Contains a number identifying the data stream type of the file or directory:

- 0 Primary Data Stream (DOS)
- 1 Macintosh Resource Fork
- 2 FRAM Extra Data Fork

fileHandle

Points to the file handle.

Remarks

`fileHandle` is not valid in the `_PRE_` open. If the file was successfully opened or created by the file system, it should be valid in the `_POST_` open.

All other fields are valid in the `_PRE_` open because they are fields that must be specified by the client to open the file. (Of course, the client does not specify the file handle.) You can get a pretty good idea as to which fields are valid by looking at the coordinating request/reply NCP structures.

Generally, items found in the request NCP structures are provided by the client and will be valid in the `_PRE_` hook. Items to be returned to the client are not valid until the `_POST_`.

PurgeDeletedCallBackStruct

Contains information about a purge deleted operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    volume ;
    LONG    dirBase ;
    LONG    toBePurgedDirBase ;
    LONG    nameSpace ;
} PurgeDeletedCallBackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the directory from which the entry is to be purged.

toBePurgedDirBase

Contains the directory base (number) that was generated while scanning for deleted files.

nameSpace

Contains the name space of the file or directory:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

RenameMoveEntryCallbackStruct

Contains information about a rename or move operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    volume ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    pathComponentCount ;
    LONG    nameSpace ;
    LONG    attributeMatchBits ;
    LONG    subDirsOnlyFlag ;
    LONG    newDirBase ;
    BYTE    *newPathString ;
    LONG    originalNewCount ;
    LONG    compatibilityFlag ;
    LONG    allowRenamesToMyselfFlag ;
} RenameMoveEntryCallbackStruct;
```

Fields

connection

Specifies the connection number of the entity requesting the operation.

task

Specifies the task number of the entity requesting the operation.

volume

Specifies the number of the volume that the directory entry is on.

dirBase

Specifies the directory base (directory number) of the file or directory.

pathString

Specifies the internal path string of the file or directory.

pathComponentCount

Specifies the number of components in the path.

nameSpace

Specifies the name space of the file or directory:

0 DOS

- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

attributeMatchBits

Specifies a bit mask of the file attributes that are affected by this operation. The first byte of the file attributes mask is as follows (see “File Attributes” on page 127):

- 0 Read Only
- 1 Hidden
- 2 System
- 3 Execute Only
- 4 Subdirectory
- 5 Archive
- 6 Undefined
- 7 Share

subDirsOnlyFlag

Specifies whether this operation is being done on a subdirectory:

TRUE Subdirectory

newDirBase

Specifies the new directory base for the entry.

newPathString

Specifies the destination path for the directory or file.

originalNewCount

Specifies the path count for the new path string.

compatibilityFlag

Specifies whether DOS 3.x locking compatibility is to be used:

TRUE Locking compatibility should be used

allowRenamesToMyselfFlag

Specifies whether this entry could be renamed to itself:

TRUE Can be renamed to itself

RenameNSEntryCallbackStruct

Contains information about a rename name space entry operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    task ;
    LONG    volume ;
    LONG    dirBase ;
    BYTE    *pathString ;
    LONG    pathComponentCount ;
    LONG    nameSpace ;
    LONG    matchBits ;
    BYTE    *newName ;
} RenameNSEntryCallbackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

task

Contains the task number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (directory number) of the file or directory.

pathString

Contains the NetWare-internal path string of the file or directory.

pathComponentCount

Contains the number of components in the path.

nameSpace

Contains the name space of the file or directory:

0 DOS

1 MACINTOSH

2 NFS

3 FTAM

4 LONG

5 NT

matchBits

Contains a bit mask of the file attributes that are affected by this operation. That is, entries that have file attributes matching this bit mask are affected. For more about the file attributes mask, see [“File Attributes” on page 127](#). The bits of the first bytes of the file attributes mask is defined as follows:

- 0 Read Only
- 1 Hidden
- 2 System
- 3 Execute Only
- 4 Subdirectory
- 5 Archive
- 6 Undefined
- 7 Share

newName

Contains the new name of the name space entry.

SalvageDeletedCallbackStruct

Contains information about a salvage deleted operation

Service: File System Monitoring

Defined In: nwfshook.h

Structure

```
typedef struct {
    LONG    connection ;
    LONG    volume ;
    LONG    dirBase ;
    LONG    toBeSalvagedDirBase ;
    LONG    nameSpace ;
    BYTE    *newName ;
} SalvageDeletedCallbackStruct;
```

Fields

connection

Contains the connection number of the entity requesting the operation.

volume

Contains the number of the volume that the directory entry is on.

dirBase

Contains the directory base (number) in which the entry is to be recovered to.

toBeSalvagedDirBase

Contains the directory base (number) that was generated while scanning for deleted files. This number is not the directory base that the file would be salvaged to (see `dirBase`, above).

nameSpace

Contains the name space of the file or directory:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

newName

Contains the name that the entry is to have after it is salvaged.

Name space allows NetWare servers to store files in formats compatible with a workstation's local file system. For example, installing the Macintosh name space allows Macintosh workstations to use Macintosh file conventions when working with network files. Although NetWare's primary name space is DOS, NetWare also supports name spaces for Macintosh, NFS, FTAM, and LONG files (OS/2 and 32-bit Windows).

Name space provides a generic interface to name space entries and associated data streams. After the NLM is loaded on a server, support for the name space must be enabled on a volume-by-volume basis. Name space entry information can include the entry's name, its attributes, significant dates and times, the owner ID, and so on.

Name space provides access to three types of data:

- ♦ primary data—available no matter which name space you are using
- ♦ specific data—specific to the name space you are using
- ♦ actual file data

This section describes the following name space features:

- ♦ [Naming Conventions \(page 427\)](#)
- ♦ [Default Name Space \(page 428\)](#)
- ♦ [Primary Entry Information \(page 428\)](#)
- ♦ [Name Space Specific Information \(page 430\)](#)
- ♦ [Long to DOS Conversions \(page 432\)](#)
- ♦ [General Name Space Functions \(page 434\)](#)

16.1 Naming Conventions

NetWare currently supports five name spaces, identified by constant names and associated numeric values:

0	NW_NS_DOS	DOS names can have up to eight upper-case characters followed by a period and up to three more upper-case characters.
1	NW_NS_MAC	Macintosh names can be up to 32 characters long including all upper and lower case printable characters, with the exception of the colon.
2	NW_NS_NFS	NFS names can be up to 256 mixed-case characters long.
3	NW_NS_FTAM	FTAM names can be up to 256 lower-case characters long.
4	NW_NS_LONG	LONG names can be up to 255 mixed-case characters long. LONG names can be used with OS/2 and any 32-bit Windows system.

DOS names remain the same in a LONG environment. NetWare uses a shortening algorithm to convert long names for use in a DOS environment. To avoid ambiguous names, this algorithm may designate a DOS file name that doesn't match the first eight characters of the long name.

16.2 Default Name Space

The CLib standard file system functions, such as ANSI fopen or POSIX open, use DOS as the default name space for the input path and filename parameters. The default name space for output path and filename parameters is also DOS. To send and receive parameters in a namespace other than DOS, you must set the current and target name space with the following functions:

SetCurrentNameSpace	Sets the name space for the paths and filenames sent to the server.
SetTargetNameSpace	Sets the name space for the paths and filenames returned by the server.

The cross platform NLM functions, which do not allow you to specify a name space, always use the DOS name space. The functions in the name space group allow you to specify a name space.

The cross platform client functions, which do not allow you to specify a name space, use the Long name space if the volume supports it. If the volume does not support the Long name space, the function uses the DOS name space. The functions in the name space group allow you to specify a name space.

16.3 Primary Entry Information

As the primary NetWare name space, the DOS name space performs a special role in the NetWare file system. All entries are represented in the DOS name space no matter what name space actually "owns" them. Consequently, if you create an entry in a name space other than DOS, you can still access the primary entry information from the DOS name space (see [“Primary Entry Information Functions” on page 430](#)).

This primary NetWare information is extended beyond DOS to accommodate Macintosh data, including information such as the number of data streams (forks) and extended attributes (Finder information).

In addition to letting you read an entry’s primary information in the DOS name space, Name Space Services enable you to read and modify this information in the name space that the entry was created in. The primary information in the owning name space varies little from what appears in the DOS name space. However, it does include the file’s long name, which isn’t available in the DOS name space.

Primary name space information includes the following items:

- ◆ Entry name
- ◆ Entry attributes
- ◆ Space allocation
- ◆ Data stream sizes
- ◆ Dates and time of events
- ◆ Inherited rights mask
- ◆ Extended attribute data
- ◆ Reference ID
- ◆ Volume Number

NW_ENTRY_INFO (page 571) contains primary name space information. The structure is filled in by **NWGetNSInfo** (page 481) or **NWScanNSEntryInfo** (page 533). Requests for primary name space information are accompanied by a return information mask, which allows you to specify which portions of **NW_ENTRY_INFO** (page 571) you want filled in. The following table shows which fields in **NW_ENTRY_INFO** (page 571) are affected by bit flags in the return information mask.

Table 16-1 Return Information Mask

Value	Constant	Affected Fields
0x0001L	IM_ENTRY_NAME	nameLength entryName
0x0002L	IM_SPACE_ALLOCATED	spaceAlloc
0x0004L	IM_ATTRIBUTES	attributes flags
0x0008L	IM_SIZE	dataStreamSize
0x0010L	IM_TOTAL_SIZE	totalStreamSize
0x0020L	IM_EA	EADataSize EAKeyCount EAKeySize
0x0040L	IM_ARCHIVE	archiveTime archiveDate archiveID
0x0080L	IM_MODIFY	modifyTime modifyDate modifierID lastAccessDate
0x0100L	IM_CREATION	creationTime creationDate creatorID
0x0200L	IM_OWNING_NAMESPACE	NSCreator
0x0400L	IM_DIRECTORY	dirEntNum DosDirNum volNumber
0x0800L	IM_RIGHTS	inheritedRightsMask

16.3.1 Primary Entry Information Functions

These functions deal with primary entry information for a name space.

NWAllocTempNSDirHandle2	Allocates a directory handle in a name space for the specified entry. The new directory handle doesn't need to be in the same name space as the original entry.
NWDeleteNSEntry	Erases the specified files from the server.
NWGetLongName	Reads an entry's name in the specified name space.
NWGetNSEntryInfo	Returns primary information for a name space entry.
NWNSRename	Renames a name space entry. Under NetWare® 4.x, 5.x, and 6.x, this function can rename an entry in a specific name space without affecting the name in other name spaces.
NWOpenCreateNSEntry	Creates a name space entry.
NWOpenDataStream	Opens or creates a data stream and returns a file handle to it.
NWOpenNSEntry	Opens a name space entry.
NWScanNSEntryInfo	Performs a file scan operation returning primary information for files matching the search mask.
NWSetLongName	Renames a name space entry.
NWSetNSEntryDOSInfo	Modifies the DOS information associated with an entry.

16.4 Name Space Specific Information

Name space specific information is maintained by the NLM that implements the name space. Much of this information may not be accessible as primary information. For example, huge data information is name space specific and must be returned by special requests

Consequently, name space includes specialized functions for accessing name space specific information. This approach requires a detailed understanding of the particular name space and the entry information it maintains.

Name space specific information is accessed by calling [NWReadNSInfo \(page 526\)](#) and [NWWriteNSInfo \(page 561\)](#). Both functions refer to the entry using a NetWare entry index, which is maintained as [NW_IDX \(page 583\)](#). To initialize [NW_IDX \(page 583\)](#), call [NWGetDirectoryBase \(page 461\)](#) and pass both a DOS directory entry (handle/path) and the target name space.

The following topics contain more detailed information:

- ◆ [“Name Space Entry Bit Mask” on page 431](#)
- ◆ [“Name Space Bit Mask” on page 431](#)
- ◆ [“DOS Name Space Bit Mask” on page 431](#)
- ◆ [“Name Space Specific Information Functions” on page 432](#)

16.4.1 Name Space Entry Bit Mask

NetWare uses a generic mechanism to represent the format of name space specific entry information. Query the NetWare server by calling [NWGetNSInfo \(page 481\)](#) to find the format for a particular name space. [NWGetNSInfo \(page 481\)](#) returns a set of bit masks as [NW_NS_INFO \(page 585\)](#). The structure indicates the size and arrangement of name space specific information.

16.4.2 Name Space Bit Mask

`NSInfoBitMask` in [NW_NS_INFO \(page 585\)](#) indicates all valid data items for an entry in the name space. [NWGetNSInfo \(page 481\)](#) initializes the bit masks for a specific name space and computes the value of `NSInfoBitMask`.

`NSInfoBitMask` is derived by combining the fixed and reserved masks through a logical OR operation.

After `NW_NS_INFO` is initialized, use it in subsequent calls to [NWReadNSInfo \(page 526\)](#) and [NWWriteNSInfo \(page 561\)](#) to read or modify name space specific entry information.

16.4.3 DOS Name Space Bit Mask

The interpretation of the name space bit mask depends on which name space you are querying. For example, the DOS name space defines the following bits:

Bit	Definition	Type	Order
0	Modify Name[13]	nuint8	
1	File Attributes	nuint32	Lo-Hi
2	Create Date	nuint16	Lo-Hi
3	Create Time	nuint16	Lo-Hi
4	Owner ID	nuint32	Hi-Lo
5	Archive Date	nuint16	Lo-Hi
6	Archive Time	nuint16	Lo-Hi
7	Archive ID	nuint32	Hi-Lo
8	Modify Date	nuint16	Lo-Hi
9	Modify Time	nuint16	Lo-Hi
10	Modify ID	nuint32	Hi-Lo
11	Last Accessed Date	nuint16	Lo-Hi
12	Inheritance Rights	nuint32	Lo-Hi
13	Maximum Space	nuint32	Lo-Hi
14-31	Reserved		

Under DOS, bit 0 represents the modify name. This is generally the case in other name spaces also. The modify name is read-only; don't attempt to modify it.

16.4.4 Name Space Specific Information Functions

These functions deal with name-space specific information:

NWGetDirectoryBase	Obtains a directory base for a name space entry.
NWGetNSInfo	Returns the information format for a name space.
NWNSGetMiscInfo	Obtains miscellaneous information for a name space entry.
NWReadExtendedNSInfo	Reads huge information for an entry.
NWReadNSInfo	Reads name space-specific information for an entry.
NWWriteExtendedNSInfo	Modifies huge information for an entry.
NWWriteNSInfo	Modifies name space-specific information for an entry.

16.5 Long to DOS Conversions

When a file is created on the server using a long name, the server automatically generates a corresponding DOS name for the file as well. This section describes the different (basic) conventions used in automatic LONG to DOS name conversions, which vary depending on the NetWare OS version you are using:

- ◆ “NetWare 4.x” on page 432
- ◆ “NetWare 5.x and 6.x” on page 434

NOTE: Since there are many circumstances in which the generated name varies (depending on the file names that already exist in the directory), you should never assume that the generated DOS name is equal to a predictable value.

For NetWare 5.x and 6.x, the algorithms are slightly more complex than the examples documented here. You might see slightly different behaviors on these more recent NetWare versions, especially if you use 8-bit ASCII characters. Also, the NSS and traditional file systems might generate slightly different names in many situations.

16.5.1 NetWare 4.x

The NetWare 4.x OS has a convention for shortening long names without periods in the first eight characters and another slightly different convention for shortening long names that have periods in the first eight characters.

If a long name has no periods, the first eight valid DOS characters become the shortened DOS name. Spaces between words of the long name are omitted. A file extension (if there is one) is retained, up to three letters.

Duplicate short names are resolved by replacing letters of the short name (not the extension) with ascending zero-based decimal numeric digits, beginning with the final letter. If necessary, an increasing number of final letters are replaced, always starting with a set of zeros. The following table illustrates the scheme:

This Is The First Long File	THISISTH
-----------------------------	----------

This Is The Second Long File	THISIST0
This Is The Third Long File	THISIST1
This Is The Fourth Long File	THISIST2
(And so on)	(And so on)
This Is The Eleventh Long File	THISIST9
This Is The Twelfth Long File	THISIS00
This Is The Thirteenth Long File	THISIS01
This Is The Fourteenth Long File	THISIS02
(And so on)	(And so on)
This Is The 112th Long File	THISI000
This Is The 113th Long File	THISI001

IMPORTANT: If one or more files are deleted, subsequent duplicate short names re-use the deleted names in ascending order before new short names are generated. For example, in the table above if "This Is The Fourth Long Name" and "This Is The Twelfth Long Name" were deleted, the next two files with initial letters "THISISTH" would be shortened to "THISISH2" and "THISIS00" before "THISI002" were generated.

If the eighth character of the long name is already a number, duplicate file naming begins with that number unless it is already used. For example, files in the same directory would be shortened as follows:

This is a 1 time offer	THISISA1
This is a 1 time deal	THISISA2
This is a 2-day tour	THISISA3
This is a 2-week tour	THISISA4
We have a 2-day pass	WEHAVEA2
We have a 2-week pass	WEHAVEA3
We have a 2-month pass	WEHAVEA4

If a long name contains a period prior to the first eight letters, the letters preceding the first period are the shortened name, and the first three letters following the final period become a file extension. Duplicate long names are shortened by adding a zero to the first duplication, two zeros to the second, and so on until letters and appended zeros make up eight characters. The next duplication begins a counting process by replacing the final zero with the digit 1.

This.File.Is.Long	THIS.LON
This.File.Is.Also.Long	THIS0.LON
This.File.Is.Really.Long	THIS00.LON
This.File.Is.Very.Long	THIS000.LON

This.File.Is.Too.Long	THIS0000.LON
This.File.Is.Much.Too.Long	THIS0001.LON
This.File.Is.Way.Too.Long	THIS0002.LON

Again, if a file is deleted, the next duplicate file is assigned the short name of the deleted file before any new short names are generated.

16.5.2 NetWare 5.x and 6.x

With NetWare 5.x and 6.x OS long names are shortened into DOS style shorter names in a consistent way that has very little variation. The first six characters are retained for four files, followed by a tilde then the digits 1 through 4. Any spaces in the first six characters are replaced with underscores. Starting with the fifth duplicate file name, only the first two characters are retained. The next four characters are replaced with random hexadecimal digits, followed by a tilde and a zero. The following table illustrates:

Long File Name	LONG_F~1
Long File Names	LONG_F~2
Long File Naming	LONG_F~3
Long File Named	LONG_F~4
Long File Name and Time	LO4104~0
Long File Name and Number	LOC5EB~0
Long File Name and Date	LO7A0D~0

If the long file name contains a period in the first six characters, the first four duplicate file names are shortened to the characters preceding the first period, followed by a tilde and the digits 1 through 4. The first three characters following the final period are retained as a file extension. Starting with the fifth file, random numbers are generated as explained above. The following table illustrates the renaming:

File.With.Internal.Period	FILE~1.PER
File.With.Another.Internal.Period	FILE~2.PER
File.With.Third.Internal.Period	FILE~3.PER
File.With.Fourth.Internal.Period	FILE~4.PER
File.With.Fifth.Internal.Period	FI58C4~0.PER
File.With.Sixth.Internal.Period	FIE95F~0.PER
File.With.Seventh.Internal.Period	FI416E~0.PER

16.6 General Name Space Functions

These functions return general information concerning name spaces.

NWGetNSLoadedList	Returns a list of numerals identifying the name spaces loaded on a particular volume.
NWGetOwningNameSpace	Returns the name space that created the specified directory entry.
NWGetNSPath	Returns the full path for an entry in a specified name space. (For name spaces that use long names, a complete entry path could potentially require a very large amount of space.)
NWNSGetDefaultNS	Returns the default name space.

This documentation describes common tasks associated with Name Space.

17.1 Accessing Huge Name Space Information

The huge information bit mask indicates large data items (between 256 and 65,535 bytes) associated with a name space entry. Call [NWReadExtendedNSInfo \(page 524\)](#) and [NWWriteExtendedNSInfo \(page 559\)](#) to access huge information. An operation on huge data must include the huge information bit mask for the name space, the length of the huge data, and a huge state information variable. This last value is maintained by the server and is used to coordinate the transmission of huge data.

Name Space Functions

18

This documentation alphabetically lists the Name Space functions and describes their purpose, syntax, parameters, and return values.

Get* and Set* Functions contains the following functions:

[GetDataStreamName \(page 440\)](#)

[GetNameSpaceName \(page 442\)](#)

[SetCurrentNameSpace \(page 444\)](#)

[SetTargetNameSpace \(page 446\)](#)

18.1 Get* and Set* Functions

Click on any function name in the table of contents to view the purpose, syntax, parameters, and return values for that function.

- ♦ [“GetDataStreamName” on page 440](#)
- ♦ [“GetNameSpaceName” on page 442](#)
- ♦ [“SetCurrentNameSpace” on page 444](#)
- ♦ [“SetTargetNameSpace” on page 446](#)

GetDataStreamName

Returns information about data streams

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>

int GetDataStreamName (
    int     volume,
    BYTE    dataStream,
    char    *dataStreamName,
    int     *numberOfDataStreams);
```

Parameters

volume

(IN) Specifies the number of the volume for which the data stream name is desired.

dataStream

(IN) Specifies the number of the data stream whose name is desired.

dataStreamName

(OUT) Points to the ASCII name of the data stream.

numberOfDataStreams

(OUT) Points to the number of data streams supported by the server.

Return Values

This function returns TRUE if the name space that defines the specified data stream is loaded on the volume. It returns FALSE if support is not loaded. If the data stream does not exist, this function returns a value of -1.

Remarks

The name of the specified data stream is returned, as well as the total number of data streams available. The function return also indicates whether the specified data stream has support on the volume.

The `dataStream` parameter is a data stream number. The defined data streams follow:

0	Primary Data Stream (corresponds to DOS)
1	Macintosh Resource Fork
2	FTAM Extra Data Fork

GetNameSpaceName

Returns the name of a specified name space and the number of name spaces currently supported by NetWare

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>

int GetNameSpaceName (
    int     volume,
    LONG    nameSpace,
    char    *name,
    int     *numberOfNameSpace);
```

Parameters

volume

(IN) Specifies the volume for which name space information is desired.

nameSpace

(IN) Specifies the number of the name space whose name is desired (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

name

(OUT) Points to the name of the name space in ASCII string (buffer length should be 32 bytes).

numberOfNameSpace

(OUT) Points to the number of name spaces currently supported by NetWare.

Return Values

-1	Specified name space does not exist.
0	Name space driver is not loaded.
1	Name space driver is loaded but is not supported on the specified volume.
2	Name space driver is loaded and supported on the specified volume.

Remarks

The five name spaces that are currently available are:

0	DOS
1	MACINTOSH
2	NFS
3	FTAM
4	LONG
5	NT

NOTE: For NSS volumes, `GetNameSpaceName` returns 2 (name space loaded and supported) for only the DOS and LONG name spaces. For the NFS and MAC name spaces, it just returns 1 (name space is loaded but not supported) on NSS volumes. These results conflict with the name space information displayed by the `VOLUMES` command. For the correct information on NSS volumes, please use the [NWGetNSLoadedList \(NLM\)](#) function.

See Also

[FEGetOriginatingNameSpace \(page 85\)](#), [SetCurrentNameSpace \(page 444\)](#), [SetTargetNameSpace \(page 446\)](#)

SetCurrentNameSpace

Sets the name space that is to be used for parsing paths that are input to server functions

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnamspc.h>

BYTE SetCurrentNameSpace (
    BYTE    newNameSpace);
```

Parameters

newNameSpace

(IN) Specifies the new name space (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

Return Values

Returns the old name space if successful. If the specified name space is not valid or is not supported on the current working volume (CWV) and current working directory (CWD), returns error code 255 and NWErrno is set to ERR_INVALID_PATH.

Remarks

SetCurrentNameSpace sets the name space to be used by the current thread group for parsing paths. This name space is used by this thread group for paths input to subsequent calls to functions from the NetWare API (until changed by another call to this function).

SetTargetNameSpace sets the name space for output from subsequent calls to functions from the NetWare API.

If you change the current name space to a non-DOS name space, CLIB will uppercase the names of newly created files and directories by default. To modify this behavior, call [UseAccurateCaseForPaths \(page 326\)](#).

See Also

[FEGetOriginatingNameSpace \(page 85\)](#), [GetNameSpaceName \(page 442\)](#), [SetTargetNameSpace \(page 446\)](#), [UseAccurateCaseForPaths \(page 326\)](#)

Example

```
#include <nwnspace.h>
BYTE oldNameSpace;
BYTE newNameSpace;

oldNameSpace=SetCurrentNameSpace (newNameSpace);
```

SetTargetNameSpace

Sets the target name space that is to be returned by server functions

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>

BYTE SetTargetNameSpace (
    BYTE    newNameSpace);
```

Parameters

newNameSpace

(IN) Specifies the new name space that is to become the target name space (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

Return Values

Returns the old target name space.

Remarks

SetTargetNameSpace sets the target name space to be used by the current thread group. This name space is used by this thread group for paths *output* from all subsequent NetWare API functions.

SetCurrentNameSpace sets the name space for input to subsequent calls to functions from the NetWare API.

See Also

[FEGetOriginatingNameSpace](#) (page 85), [SetCurrentNameSpace](#) (page 444)

18.2 NWA* through NWI* Functions

Click on any function name in the table of contents to view the purpose, syntax, parameters, and return values for that function.

- ♦ [“NWAddTrusteeToNSDirectory”](#) on page 448
- ♦ [“NWAllocTempNSDirHandle2”](#) on page 451

- ◆ “NWAllocTempNSDirHandle2Ext” on page 453
- ◆ “NWDeleteNSEntry” on page 455
- ◆ “NWDeleteNSEntryExt” on page 457
- ◆ “NWDeleteTrusteeFromNSDirectory” on page 459
- ◆ “NWGetDirectoryBase” on page 461
- ◆ “NWGetDirectoryBaseExt” on page 464
- ◆ “NWGetHugeNSInfo” on page 466
- ◆ “NWGetLongName” on page 468
- ◆ “NWGetLongNameExt” on page 470
- ◆ “NWGetNameSpaceEntryName” on page 472
- ◆ “NWGetNSEntryInfo” on page 474
- ◆ “NWGetNSEntryInfoExt” on page 477
- ◆ “NWGetNSFileDirEntryNumber” on page 479
- ◆ “NWGetNSInfo” on page 481
- ◆ “NWGetNSInfo (NLM)” on page 483
- ◆ “NWGetNSLoadedList” on page 485
- ◆ “NWGetNSLoadedList (NLM)” on page 487
- ◆ “NWGetNSPath” on page 489
- ◆ “NWGetNSPathExt” on page 491
- ◆ “NWGetOwningNameSpace” on page 493
- ◆ “NWIsLNSSupportedOnVolume” on page 495

NWAddTrusteeToNSDirectory

Adds a trustee to the trustee list in a directory for the specified name space.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE NWAddTrusteeToNSDrectory (
    NWCONN_HANDLE      conn,
    nuint8              namSpc,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint32             trusteeID,
    nuint8              rightsMask);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

namSpc

(IN) Specifies the name space for the resulting trustee (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dirHandle

(IN) Specifies the directory handle associated with the desired directory path under the specified name space (0 if `path` contains the complete path, including the volume name).

path

(IN) Points to the absolute path (or a path relative to the directory handle) of the directory to which a trustee is being added.

trusteeID

(IN) Specifies the object ID for the object being added as a trustee.

rightsMask

(IN) Specifies the access rights mask the new trustee is being granted (see “Trustee Rights” on page 124).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x8990	NO_FILES_AFFECTED_READ_ONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x8999	DIRECTORY_FULL
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FC	NO_SUCH_OBJECT
0x89FD	BAD_STATION_NUMBER
0x89FF	HARDWARE_FAILURE

Remarks

If the object is already a trustee for the specified directory, the current access mask of the trustee is replaced by the value contained in the `trusteeID` parameter. Otherwise, the object is added as a trustee to the directory and given a rights mask equal to the `trusteeID` parameter.

If you are using an NDS object name as the trustee name, call [NWDSMapNameToID](#) to return the value to pass to `trusteeID`.

To modify a trustee rights list, the requesting workstation must have access control rights to the directory or to a parent of the directory.

The object must be static. If the object is dynamic, `NWAddTrusteeToNSDirectory` will return an error.

NCP Calls

0x2222 22 13 Add Trustee To Directory
0x2222 22 39 Trustee Add Ext
0x2222 23 17 Get File Server Information

0x2222 87 10 Add Trustee Set To File Or Subdirectory

See Also

[NWAddTrustee \(page 153\)](#), [NWAddTrusteeToDirectory \(page 158\)](#), [NWDeleteTrustee \(page 177\)](#),
[NWDeleteTrusteeFromDirectory \(page 181\)](#), [NWDeleteTrusteeFromNSDirectory \(page 459\)](#),
[NWScanNSDirectoryForTrustees \(page 530\)](#)

NWAllocTempNSDirHandle2

Assigns a temporary directory handle in the specified name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWAllocTempNSDirHandle2 (
    NWCONN_HANDLE      conn,
    nuint8             dirHandle,
    const nstr8 N_FAR  *path,
    nuint8             nameSpc,
    pnuint8            newDirHandle,
    nuint8             newNameSpace);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWAllocTempNSDirHandle2
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   const path : pnstr8;
   namSpc : nuint8;
   newDirHandle : pnuint8;
   newNameSpace : nuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle through which to attach.

dirHandle

(IN) Specifies the directory handle associated with the desired directory path.

path

(IN) Points to an absolute path, (or relative if `dirHandle` is non-zero), with which `dirHandle` is to be associated.

namSpc

(IN) Specifies the name space of the `dirHandle/path` combination (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

newDirHandle

(OUT) Points to the new directory handle.

newNameSpc

(IN) Specifies the name space to be used for the new directory handle (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

Return Values

These are common return values; see [Return Values \(*Return Values for C*\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89BF	INVALID_NAME_SPACE

NCP Calls

0x2222 23 17 Get File Server Information
0x2222 87 06 Obtain File or Subdirectory Information
0x2222 87 12 Allocate Short Directory Handle

NWAllocTempNSDirHandle2Ext

Assigns a temporary directory handle in the specified name space, using UTF-8 strings

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWAllocTempNSDirHandle2Ext (
    NWCONN_HANDLE      conn,
    nuInt8              dirHandle,
    const nstr8 N_FAR  *path,
    nuInt8              nameSpc,
    pnuInt8             newDirHandle,
    nuInt8              newNameSpace);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle through which to attach.

dirHandle

(IN) Specifies the directory handle associated with the desired directory path.

path

(IN) Points to an absolute path, (or relative if `dirHandle` is non-zero), with which `dirHandle` is to be associated. The characters in the string must be UTF-8.

namSpc

(IN) Specifies the name space of the `dirHandle/path` combination (see [Section 20.5](#), “Name Space Flag Values,” on page 595).

newDirHandle

(OUT) Points to the new directory handle.

newNameSpc

(IN) Specifies the name space to be used for the new directory handle (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89BF	INVALID_NAME_SPACE

NCP Calls

0x2222 23 17 Get File Server Information
0x2222 87 06 Obtain File or Subdirectory Information
0x2222 87 12 Allocate Short Directory Handle
0x2222 89 12 Allocate Short Directory Handle

See Also

[NWAllocTempNSDirHandle2 \(page 451\)](#)

NWDeleteNSEntry

Erases the specified files from the server

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWDeleteNSEntry (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *fileName,
    nuint8              nameSpace,
    nuint16             searchAttr);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWDeleteNSEntry
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const fileName : pnstr8;
   nameSpace : nuint8;
   searchAttr : nuint16
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare connection handle.

dirHandle

(IN) Specifies the directory handle on which files to be deleted currently reside.

fileName

(IN) Points to an absolute path (or relative if `dirHandle` is non-zero) that cannot exceed 255 characters in length.

nameSpace

(IN) Specifies the name space of `dirHandle/filePath` (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

searchAttr

(IN) Specifies the file attributes to use in finding the file (see [Section 20.8, “Search Attributes Values,”](#) on page 597).

Return Values

These are common return values; see [Return Values \(*Return Values for C*\)](#) for more information.

0x0000	SUCCESSFUL
0x898A	NO_DELETE_PRIVILEGES
0x898D	SOME_FILES_AFFECTED_IN_USE
0x898E	NO_FILES_AFFECTED_IN_USE
0x898F	SOME_FILES_AFFECTED_READ_ONLY
0x8990	NO_FILES_AFFECTED_READ_ONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	NO_FILES_FOUND_ERROR

Remarks

`dirHandle` must exist in the designated name space.

If a file has the immediate purge attribute set, the file cannot be recovered.

NCP Calls

0x2222 68 Erase File

0x2222 87 08 Delete A File Or Subdirectory

See Also

[NWIntEraseFiles](#) (page 218), [NWOpenCreateNSEntry](#) (page 508), [NWRecoverDeletedFile](#) (page 49)

NWDeleteNSEnterExt

Erases the specified files from the server, using UTF-8 strings

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 and later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWDeleteNSEnter (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *fileName,
    nuint8              nameSpace,
    nuint16             searchAttr);
```

Parameters

conn

(IN) Specifies the NetWare connection handle.

dirHandle

(IN) Specifies the directory handle on which files to be deleted currently reside.

fileName

(IN) Points to an absolute path (or relative if `dirHandle` is non-zero) that cannot exceed 255 characters in length. The characters in the string must be UTF-8.

nameSpace

(IN) Specifies the name space of `dirHandle/filePath` (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

searchAttr

(IN) Specifies the file attributes to use in finding the file (see [Section 20.8, “Search Attributes Values,”](#) on page 597).

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x88F0	UTF8_CONVERSION_FAILED
0x898A	NO_DELETE_PRIVILEGES
0x898D	SOME_FILES_AFFECTED_IN_USE
0x898E	NO_FILES_AFFECTED_IN_USE
0x898F	SOME_FILES_AFFECTED_READ_ONLY
0x8990	NO_FILES_AFFECTED_READ_ONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	NO_FILES_FOUND_ERROR

Remarks

`dirHandle` must exist in the designated name space.

If a file has the immediate purge attribute set, the file cannot be recovered.

NCP Calls

0x2222 68 Erase File

0x2222 87 08 Delete A File Or Subdirectory

0x2222 89 08 Delete A File Or Subdirectory

See Also

[NWDeleteNSEntry](#) (page 455)

NWDeleteTrusteeFromNSDirectory

Removes a trustee from a directory trustee list in the specified name space.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: File System

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWDeleteTrusteeFromNSDirectory (
    NWCONN_HANDLE      conn,
    nuint8              namSpc,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *dirPath,
    nuint32             objID);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

namSpc

(IN) Specifies the name space in which the trustee resides (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dirHandle

(IN) Specifies the NetWare directory handle for the directory whose trustee list is being modified (zero if the `path` parameter points to the complete path, including the volume name).

dirPath

(IN) Points to an absolute path (or a path relative to the `dirHandle` parameter) specifying the directory from which the trustee is being removed.

objID

(IN) Specifies the object ID for the trustee being deleted.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000 SUCCESSFUL

Remarks

NWDeleteTrusteeFromNSDirectory revokes the rights for a trustee in a specific directory. The requesting workstation must have access control rights in the directory or in a parent directory to delete a trustee.

Deleting the explicit assignment of an trustee object in a directory is not the same as assigning no rights to the object in the directory. If no rights are assigned in a directory, the object inherits the same rights it has in the parent directory.

NCP Calls

0x2222 87 11 Delete Trustee Set From File Or Subdirectory

See Also

[NWAddTrusteeToDirectory \(page 158\)](#), [NWAddTrusteeToNSDirectory \(page 448\)](#),
[NWDeleteTrusteeFromDirectory \(page 181\)](#), [NWParseNetWarePath \(page 622\)](#),
[NWScanDirectoryForTrustees2 \(page 262\)](#), [NWScanNSDirectoryForTrustees \(page 530\)](#)

NWGetDirectoryBase

Retrieves information used in further calls to the name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWGetDirectoryBase (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              dstNamSpc,
    NW_IDX N_FAR       *idxStruct);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetDirectoryBase
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   const path : pnstr8;
   dstNamSpc : nuint8;
   Var idxStruct : NW_IDX
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to search.

path

(IN) Points to a valid DOS path (pointing to a directory or a file).

dstNamSpc

(IN) Specifies the destination name space (see [Section 20.5, “Name Space Flag Values,”](#) on [page 595](#)).

idxStruct

(OUT) Points to NW_IDX.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89BF	INVALID_NAME_SPACE

Remarks

The `path` parameter must be upper case if `dirHandle` contains a DOS name space directory handle.

The `path` and `dirHandle` parameters must match the `dstNamSpc` parameter.

NetWare uses the `idxStruct` parameter as an index to quickly locate a directory entry (file or directory). It is required as a calling parameter to other functions and should not be modified by the application.

NCP Calls

- 0x2222 22 3 Get Directory Effective Rights
- 0x2222 22 19 Allocate Temporary Directory Handle
- 0x2222 22 20 Free Directory Handle
- 0x2222 23 15 Scan Files
- 0x2222 23 17 Get File Server Information
- 0x2222 68 File Erase
- 0x2222 87 2 Scan First
- 0x2222 87 3 Scan Next
- 0x2222 87 8 Delete Entry
- 0x2222 87 12 Allocate Directory Handle
- 0x2222 87 22 Generate Directory Base And Volume Number

See Also

[NWNSGetMiscInfo \(page 500\)](#), [NWReadExtendedNSInfo \(page 524\)](#), [NWReadNSInfo \(page 526\)](#), [NWWriteExtendedNSInfo \(page 559\)](#), [NWWriteNSInfo \(page 561\)](#)

NWGetDirectoryBaseExt

Retrieves information used in further calls to the name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWGetDirectoryBaseExt (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              dstNamSpc,
    NW_IDX N_FAR       *idxStruct);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to search.

path

(IN) Points to a valid DOS path (pointing to a directory or a file). The characters in the string must be UTF-8.

dstNamSpc

(IN) Specifies the destination name space (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

idxStruct

(OUT) Returns a filled in NW_IDX structure.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89BF	INVALID_NAME_SPACE

Remarks

The `path` parameter must be upper case if `dirHandle` contains a DOS name space directory handle.

The `path` and `dirHandle` parameters must match the `dstNamSpc` parameter.

NetWare uses the `idxStruct` parameter as an index to quickly locate a directory entry (file or directory). It is required as a calling parameter to other functions and should not be modified by the application.

NCP Calls

0x2222 22 3 Get Directory Effective Rights
0x2222 22 19 Allocate Temporary Directory Handle
0x2222 22 20 Free Directory Handle
0x2222 23 15 Scan Files
0x2222 23 17 Get File Server Information
0x2222 68 File Erase
0x2222 87 2 Scan First
0x2222 87 3 Scan Next
0x2222 87 8 Delete Entry
0x2222 87 12 Allocate Directory Handle
0x2222 87 22 Generate Directory Base And Volume Number
0x2222 89 22 Generate Directory Base And Volume Number

See Also

[NWGetDirectoryBase \(page 461\)](#)

NWGetHugeNSInfo

Gets extended (huge) NS information for the entry specified by `volNum`, `nameSpace` and `dirBase`

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>

int NWGetHugeNSInfo (
    BYTE    volNum,
    BYTE    nameSpace,
    LONG    dirBase,
    LONG    hugeInfoMask,
    BYTE    *hugeStateInfo,
    BYTE    *hugeData,
    LONG    *hugeDataLen,
    BYTE    *nextHugeStateInfo);
```

Parameters

volNum

(IN) Specifies the volume number for which huge NS information is to be obtained.

nameSpace

(IN) Specifies the name space for which huge information is being returned (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dirBase

(IN) Specifies the directory base (or number) for the entry for which information is being obtained.

hugeInfoMask

(IN) Specifies the bit map that indicates which types of information the user wants returned. (Corresponds to the `extendedBitMask` in the `NW_NS_INFO` struct that can be retrieved by calling `NWQueryNSInfoFormat`.)

hugeStateInfo

(IN) Points to the first time calling this function, this should be set to zeroes. On succeeding calls, the `nextHugeStateInfo` should be passed in this parameter.

hugeData

(OUT) Points to data returned as specified in the `hugeInfoMask`.

hugeDataLen

(OUT) Points to length of the huge data the name space returned.

nextHugeStateInfo

(OUT) Points to huge state information that should be passed in on the next call to this function. It is zero-filled when reading is done.

Return Values

ESuccess or NetWare errors

Remarks

This function retrieves extended NS information for `nameSpace` and returns it in `hugeData`.

See Also

[NWGetDirBaseFromPath \(page 610\)](#), [NWQueryNSInfoFormat \(page 522\)](#), [NWSetHugeNSInfo \(page 544\)](#)

NWGetLongName

Retrieves a filename for the specified name space.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWGetLongName (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              srcNamSpc,
    nuint8              dstNamSpc,
    pnstr8              longName);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetLongName
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   const path : pnstr8;
   srcNamSpc : nuint8;
   dstNamSpc : nuint8;
   longName : pnstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to scan. It can be 0 if path contains a fully specified path.

path

(IN) Points to a valid path. This can either be a fully specified path (vol:path), or it can be relative to dirHandle.

srcNamSpc

(IN) Specifies the name space referred to by dirHandle/path (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dstNamSpc

(IN) Specifies the name space for the return name (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

longName

(OUT) Points to a buffer returning the corresponding name space’s name (up to 256 bytes).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

longName includes only the name of the last component in the path. NWGetLongName does not translate the entire path to a new name in the designated name space.

The name returned is the same name returned by NWGetNSEntryInfo.

NCP Calls

0x2222 87 06 Obtain File or Subdirectory Information

See Also

[NWGetNSEntryInfo](#) (page 474), [NWGetNSPath](#) (page 489), [NWSetLongName](#) (page 546)

NWGetLongNameExt

Retrieves a filename for the specified name space, using UTF-8 strings

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWGetLongNameExt (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              srcNamSpc,
    nuint8              dstNamSpc,
    pnstr8              longName);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to scan. It can be 0 if `path` contains a fully specified path.

path

(IN) Points to a valid path. This can either be a fully specified path (`vol:path`), or it can be relative to `dirHandle`. The characters in the string must be UTF-8.

srcNamSpc

(IN) Specifies the name space referred to by `dirHandle/path` (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

dstNamSpc

(IN) Specifies the name space for the return name (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

longName

(OUT) Points to a buffer returning the corresponding name space's name (up to 256 bytes). The returned name is UTF-8.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

longName includes only the name of the last component in the path. NWGetLongNameExt does not translate the entire path to a new name in the designated name space.

The name returned is the same name returned by NWGetNSEntryInfoExt.

NCP Calls

0x2222 87 06 Obtain File or Subdirectory Information

0x2222 89 06 Obtain File or Subdirectory Information

See Also

[NWGetNSEntryInfoExt \(page 477\)](#), [NWGetLongName \(page 468\)](#)

NWGetNameSpaceEntryName

Returns the name of a file or directory in the specified name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>
```

```
int NWGetNameSpaceEntryName (
    BYTE    *path,
    LONG    nameSpace,
    LONG    maxNameBufferlength,
    BYTE    *nameSpaceEntryName);
```

Parameters

path

(IN) Points to the path to the file system entry to get a name space entry name.

nameSpace

(IN) Specifies the name space to get the file or directory name for (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

maxNameBufferLength

(IN) Specifies the maximum length of a name that can be stored in the buffer specified by nameSpaceEntryName.

nameSpaceEntryName

(IN) Points to a buffer in which to store the name.

Return Values

ESuccess or NetWare errors

Remarks

If you know the name of a file or directory in one name space—DOS, Macintosh, NFS—you can find out its name in other name spaces by calling NWGetNameSpaceEntryName.

The path specified in the path parameter must be in your current name space. For more information, see [Section 16.2, “Default Name Space,”](#) on page 428.

See Also

[NWSetNameSpaceEntryName \(page 549\)](#)

NWGetNSEntryInfo

Returns name space entry information for the entry referred to by the `dirHandle` and `path` combination

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWGetNSEntryInfo (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    nuint8             srcNamSpc,
    nuint8             dstNamSpc,
    nuint16            searchAttrs,
    nuint32            retInfoMask,
    NW_ENTRY_INFO N_FAR *entryInfo);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetNSEntryInfo
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   const path : pnstr8;
   srcNamSpc : nuint8;
   dstNamSpc : nuint8;
   searchAttrs : nuint16;
   retInfoMask : nuint32;
   Var entryInfo : NW_ENTRY_INFO
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired name space (optional).

path

(IN) Points to the valid DOS path (pointing to a directory or file).

srcNamSpc

(IN) Specifies the name space of `dirHandle/path` (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dstNamSpc

(IN) Specifies the name space for the return information (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

searchAttrs

(IN) Specifies the search attributes to use (see [Section 20.8, “Search Attributes Values,”](#) on page 597).

retInfoMask

(IN) Specifies the information to return (see [Section 20.6, “Basic Return Mask Values,”](#) on page 595).

entryInfo

(OUT) Points to `NW_ENTRY_INFO`. Only fields related to `retInfoMask` are valid.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89BF	INVALID_NAME_SPACE
0x89FF	Bad Parameter—no constant

Remarks

`dirHandle` can be zero if `path` contains the complete path, including the volume name. `dirHandle` and/or `path` contains the entry name according to `srcNamSpc`. This information is returned for `dstNamSpc`.

To request information from a server, a client sets the appropriate bit or bits of `retInfoMask` and sends a request packet to the server.

NCP Calls

0x2222 87 06 Obtain File Or Subdirectory Information

See Also

[NWGetOwningNameSpace \(page 493\)](#), [NWGetLongName \(page 468\)](#)

NWGetNSEntryInfoExt

Returns name space entry information for the specified entry, using UTF-8 strings

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 and later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWGetNSEntryInfoExt (
    NWCONN_HANDLE          conn,
    NWDIR_HANDLE           dirHandle,
    const nstr8 N_FAR      *path,
    nuint8                  srcNamSpc,
    nuint8                  dstNamSpc,
    nuint16                 searchAttrs,
    nuint32                 retInfoMask,
    NW_ENTRY_INFO_EXT N_FAR *entryInfo);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired name space (optional).

path

(IN) Points to the valid DOS path (pointing to a directory or file). The characters in the string must be UTF-8.

srcNamSpc

(IN) Specifies the name space of dirHandle/path (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

dstNamSpc

(IN) Specifies the name space for the return information (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

searchAttrs

(IN) Specifies the search attributes to use (see [Section 20.8, “Search Attributes Values,”](#) on [page 597](#)).

retInfoMask

(IN) Specifies the information to return (see [Section 20.6, “Basic Return Mask Values,”](#) on [page 595](#)).

entryInfo

(OUT) Points to NW_ENTRY_INFO_EXT. Only fields related to `retInfoMask` are valid.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89BF	INVALID_NAME_SPACE
0x89FF	Bad Parameter—no constant

Remarks

`dirHandle` can be zero if `path` contains the complete path, including the volume name. `dirHandle` and/or `path` contains the entry name according to `srcNamSpc`. This information is returned for `dstNamSpc`.

To request information from a server, a client sets the appropriate bit or bits of `retInfoMask` and sends a request packet to the server.

NCP Calls

0x2222 87 06 Obtain File Or Subdirectory Information

0x2222 89 06 Obtain File Or Subdirectory Information

See Also

[NWGetLongNameExt \(page 470\)](#)

NWGetNSFileDirEntryNumber

Returns file information for a specified file under DOS and the name space associated with the specified file handle

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwfile.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY (NWCCODE) NWGetNSFileDirEntryNumber (
    NWFIL_HANDLE fileHandle,
    nuInt8 nameSpace,
    pnuInt32 volumeNum,
    pnuInt32 directoryEntry,
    pnuInt32 dataStream);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetNSFileDirEntryNumber
  (fileHandle : NWFIL_HANDLE;
   nameSpace : nuInt8;
   volumeNum : pnuInt32;
   directoryEntry : pnuInt32;
   dataStream : pnuInt32;
  ) : NWCCODE;
```

Parameters

fileHandle

(IN) Specifies the file handle.

nameSpace

(IN) Specifies the name space associated with the `directoryEntry` parameter (see [Section 20.5, "Name Space Flag Values," on page 595](#)).

volumeNum

(OUT) Points to the volume number of the file handle.

directoryEntry

(OUT) Points to the directory entry number in the name space associated with the `nameSpace` parameter.

dataStream

(OUT) Points to the data stream number if the name space is `NW_NS_MAC`:

1 Data fork

0 Resource fork and anything else

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x0006	INVALID_HANDLE
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8988	INVALID_FILE_HANDLE

Remarks

`NWGetNSFileDirEntryNumber` returns the volume number and directory entry numbers in the name space specified by the `nameSpace` parameter.

Call the `NWGetFileDirEntryNumber` function to return the parent directory number. The `NWGetFileDirEntryNumber` allows you to specify the name space in which to return the parent directory number.

One way to create the file handle is to call the `NWOpenNSEntry` function. If you specify a long file name, the created file handle will be associated with the `LONG` name space. If a DOS file name is specified, the created file handle will be associated with the `DOS` name space.

NCP Calls

87 31 Get File Information

See Also

[NWOpenNSEntry \(page 516\)](#)

NWGetNSInfo

Returns the NW_NS_INFO structure to be used in reading and writing information to the name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWGetNSInfo (
    NWCONN_HANDLE      conn,
    const NW_IDX N_FAR *idxStruct,
    NW_NS_INFO N_FAR  *NSInfo);
```

Delphi Syntax

```
uses calwin32;

Function NWGetNSInfo
  (conn : NWCONN_HANDLE;
   const idxStruct : pNW_IDX;
   Var NSInfo : NW_NS_INFO
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

idxStruct

(IN) Points to the NW_IDX structure.

NSInfo

(OUT) Points to the NW_NS_INFO structure.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION

Remarks

NW_IDX is returned by NWNSGetMiscInfo or NWGetDirectoryBase. The dstNameSpace parameter in each function obtains the Name Space information.

NSInfo is returned for the destination name space in idxStruct.

NCP Calls

0x2222 87 23 Query NS Information Format

See Also

[NWGetDirectoryBase \(page 461\)](#), [NWNSGetMiscInfo \(page 500\)](#), [NWReadExtendedNSInfo \(page 524\)](#), [NWReadNSInfo \(page 526\)](#), [NWWriteExtendedNSInfo \(page 559\)](#), [NWWriteNSInfo \(page 561\)](#)

NWGetNSInfo (NLM)

Returns specific NS information for the entry specified by the `volNum`, `nameSpace` and `dirBase` parameters

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>

int NWGetNSInfo (
    BYTE    volNum,
    BYTE    srcNameSpace,
    BYTE    dstNameSpace,
    LONG    dirBase,
    LONG    nsInfoMask,
    BYTE    *nsSpecificInfo);
```

Parameters

volNum

(IN)

Specifies the volume number for which information is to be returned.

srcNameSpace

(IN) Specifies the name space that corresponds with the `dirBase` being passed.

dstNameSpace

(IN) Specifies name space in which the information is to be returned.

dirBase

(IN) Specifies the directory base (or number) for the entry for which information is being retrieved.

nsInfoMask

(IN) Specifies the bit map that indicates which types of information the user wants returned in the data parameter.

nsSpecificInfo

(OUT) Points to data that was asked for as indicated in the `nsInfoMask`.

Return Values

ESuccess or NetWare errors

Remarks

If the current name space is NFS, a value of 2 (for NFS) would be passed to the `srcNameSpace` parameter. However, if the returned information should be in the Macintosh name space format, a value of 1 would be passed to the `dstNameSpace` parameter.

See [“DOS Name Space Bit Mask” on page 431](#).

See Also

[NWGetDirBaseFromPath \(page 610\)](#), [NWQueryNSInfoFormat \(page 522\)](#), [NWSetNSInfo \(page 557\)](#)

NWGetNSLoadedList

Retrieves a list of the name spaces loaded for the specified volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWGetNSLoadedList (
    NWCONN_HANDLE   conn,
    nuint8           volNum,
    nuint8           maxListLen,
    pnuint8          NSLoadedList,
    pnuint8          actualListLen) ;
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetNSLoadedList
  (conn : NWCONN_HANDLE;
   volNum : nuint8;
   maxListLen : nuint8;
   NSLoadedList : pnuint8;
   actualListLen : pnuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNum

(IN) Specifies the volume number to obtain the list from.

maxListLen

(IN) Specifies the size of NSLoadedList (in bytes).

NSLoadedList

(OUT) Points to a buffer (`maxListLen` bytes).

actualListLen

(OUT) Points to the number of name spaces loaded (in bytes).

Return Values

These are common return values; see [Return Values \(*Return Values for C*\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION

Remarks

`NSLoadedList` contains a `nuint8` entry for every name space loaded on the server. The buffer for `NSLoadedList` should be at least 5 bytes long (`maxListLen` should also be at least 5 bytes).

NCP Calls

0x2222 87 24 Get Name Spaces Loaded List From Volume Number

NWGetNSLoadedList (NLM)

Retrieves a list of the name spaces that are loaded on the specified volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>

int NWGetNSLoadedList (
    BYTE    volNum,
    WORD    loadListSize,
    BYTE    *NSLoadedList,
    WORD    *returnListSize);
```

Parameters

volNum

(IN) Specifies the volume number for which to get the list of loaded name spaces.

loadListSize

(IN) Specifies the size (in bytes) of the `NSLoadedList` buffer being passed.

NSLoadedList

(OUT) Points to a buffer to hold the loaded name spaces.

returnListSize

(OUT) Points to the number of name spaces loaded.

Return Values

ESuccess or NetWare errors

Remarks

The `NSLoadedList` contains a `BYTE` entry for every name space that is loaded on the volume. The buffer for `NSLoadedList` needs to be at least `MAX_NAMESPACES` bytes long (therefore, `loadListSize` needs to be at least `MAX_NAMESPACES`). In the case where there are more name spaces loaded than there is space available in the `NSLoadedList` buffer, `returnListSize` contains the number of name spaces loaded.

See Also

[NWQueryNSInfoFormat \(page 522\)](#)

NWGetNSPath

Returns the full NetWare path for the desired name space associated with the specified path

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWGetNSPath (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint16             fileFlag,
    nuint8              srcNamSpc,
    nuint8              dstNamSpc,
    NW_NS_PATH N_FAR  *NSPath);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetNSPath
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   fileFlag : nuint16;
   srcNamSpc : nuint8;
   dstNamSpc : nuint8;
   Var NSPath : NW_NS_PATH
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired name space.

fileFlag

(IN) Specifies whether the source path ends with a file or a directory name:

0 = directory name

1 = file name

srcNamSpc

(IN) Specifies the name space used for srcPath in NSPath (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dstNamSpc

(IN) Specifies the name space for the return path (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

NSPath

(IN/OUT) Points to NW_NS_PATH.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

A full path includes the volume name. For example:

```
volume:path\path
```

If the fileFlag parameter is set to 0 (indicating a directory name is being passed) and a file name is passed, INVALID_PARAMETER will be returned. The same error will be returned if the fileFlag parameter is set to 1 (indicating a file name is being passed) and a directory name is passed.

NWGetNSPath returns only the directory path name even if a file name was passed.

On NetWare server versions 3.12 and before, NWGetNSPath will return INVALID_PATH when used to return the full path of a root file.

NCP Calls

0x2222 87 28 Get Full Path String

NWGetNSPathExt

Returns the full NetWare path for the desired name space associated with the specified path, using UTF-8 strings

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWGetNSPathExt (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint16             fileFlag,
    nuint8              srcNamSpc,
    nuint8              dstNamSpc,
    NW_NS_PATH N_FAR *NSPath);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the desired name space.

fileFlag

(IN) Specifies whether the source path ends with a file or a directory name:

0 = directory name
1 = file name

srcNamSpc

(IN) Specifies the name space used for srcPath in NSPath (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dstNamSpc

(IN) Specifies the name space for the return path (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

NSPath

(IN/OUT) Points to NW_NS_PATH.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

A full path includes the volume name. For example:

```
volume:path\path
```

If the `fileFlag` parameter is set to 0 (indicating a directory name is being passed) and a file name is passed, `INVALID_PARAMETER` will be returned. The same error will be returned if the `fileFlag` parameter is set to 1 (indicating a file name is being passed) and a directory name is passed.

`NWGetNSPathExt` returns only the directory path name even if a file name was passed.

NCP Calls

0x2222 87 28 Get Full Path String

0x2222 89 28 Get Full Path String

NWGetOwningNameSpace

Returns the owning name space for the specified directory or file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWGetOwningNameSpace (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    const nstr8 N_FAR  *path,
    pnuint8             nameSpace);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetOwningNameSpace
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   const path : pnstr8;
   namSpc : pnuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to search.

path

(IN) Points to a valid NetWare path (pointing to a directory or file).

nameSpace

(OUT) Points to the owning name space (see [Section 20.5, “Name Space Flag Values,”](#) on [page 595](#)).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

The owning name space is defined as the name space under which the entry (file or directory) was created.

Both the `dirHandle` and `path` parameters must be in the default name space.

The default name space is the name space that matches the OS and the loaded name spaces on that volume. For example, Windows 95 on a volume with LONG name space will set LONG name space as the default name space.

NCP Calls

0x2222 87 06 Obtain File or Subdirectory Information

NWIsLNSSupportedOnVolume

Queries the NetWare server and returns a nonzero if the LONG name space is supported on the target volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwmisc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWIsLNSSupportedOnVolume (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path);
```

Delphi Syntax

```
uses calwin32;

Function NWIsLNSSupportedOnVolume
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pnstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the volume whose status is being checked.

path

(IN) Points to the absolute directory path (or a path relative to the directory handle) associated with the volume whose status is being checked.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	LONG name space not supported on volume
nonzero	LONG name space supported on volume

Remarks

NWIsLNSSupportedOnVolume is called in a Windows 32-bit platform to determine whether DOS names or LONG names should be used in paths (see [Section 16.1, “Naming Conventions,” on page 427](#)).

In Windows 32-bit platforms, if a nonzero value is returned, use LONG names when calling NWCalls. On 3.11 servers and above, NWCalls expects LONG names to be used on all volumes having the LONG name space loaded.

In Windows 32-bit platforms, if the `dirHandle` or `path` parameters are invalid, 0x0000 will always be returned. Therefore, make sure the `dirHandle` and `path` parameters are valid before calling NWIsLNSSupportedOnVolume.

NCP Calls

0x2222 23 17 Get File Server Information

0x2222 23 234 Get Connection’s Task Information

18.3 NWN* through NWW* Functions

Click on any function name in the table of contents to view the purpose, syntax, parameters, and return values for that function.

- ♦ [“NWNSGetDefaultNS” on page 498](#)
- ♦ [“NWNSGetMiscInfo” on page 500](#)
- ♦ [“NWNSRename” on page 502](#)
- ♦ [“NWNSRenameExt” on page 505](#)
- ♦ [“NWOpenCreateNSEntry” on page 508](#)
- ♦ [“NWOpenCreateNSEntryExt” on page 510](#)
- ♦ [“NWOpenDataStream” on page 512](#)
- ♦ [“NWOpenNSEntry” on page 516](#)
- ♦ [“NWOpenNSEntryExt” on page 519](#)
- ♦ [“NWQueryNSInfoFormat” on page 522](#)
- ♦ [“NWReadExtendedNSInfo” on page 524](#)
- ♦ [“NWReadNSInfo” on page 526](#)
- ♦ [“NWReadNSInfoExt” on page 528](#)
- ♦ [“NWScanNSDirectoryForTrustees” on page 530](#)

- ◆ “NWScanNSEntryInfo” on page 533
- ◆ “NWScanNSEntryInfoExt” on page 536
- ◆ “NWScanNSEntryInfo2” on page 538
- ◆ “NWScanNSEntryInfoSet” on page 541
- ◆ “NWSetHugeNSInfo” on page 544
- ◆ “NWSetLongName” on page 546
- ◆ “NWSetNameSpaceEntryName” on page 549
- ◆ “NWSetNSEntryDOSInfo” on page 551
- ◆ “NWSetNSEntryDOSInfoExt” on page 554
- ◆ “NWSetNSInfo” on page 557
- ◆ “NWWriteExtendedNSInfo” on page 559
- ◆ “NWWriteNSInfo” on page 561
- ◆ “NWWriteNSInfoExt” on page 563

NWNSGetDefaultNS

Returns the default name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWNSGetDefaultNS (
    NWCONN_HANDLE      conn,
    NWDIR_HANDLE       dirHandle,
    const nstr8 N_FAR  *path,
    puint8              pbuDefaultNameSpace);
```

Delphi Syntax

```
uses calwin32;

Function NWNSGetDefaultNS
  (conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   const path : pustr8;
   pbuDefaultNameSpace : puint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory for which to return the default name space.

path

(IN) Points to a valid NetWare path (pointing to a directory or a file).

pbuDefaultNameSpace

(OUT) Points to the default name space.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8836	INVALID_PARAMETER
0x890A	NLM_INVALID_CONNECTION
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x89FF	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

Both the `dirHandle` and `path` parameters must be in the default name space.

The default name space is the name space that matches the OS and the loaded name spaces on that volume. For example, Windows 95 on a volume with LONG name space will set LONG name space as the default name space.

NCP Calls

0x2222 22 5 Get Volume Number

0x2222 22 21 Get Volume Info With Handle

0x2222 87 24 Get Name Spaces Loaded List From Volume Number

See Also

[NWGetVolumeInfoWithHandle](#), [NWGetVolumeNumber](#) (Volume Management)

NWNSGetMiscInfo

Retrieves information to be used in further calls to the name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWNSGetMiscInfo (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              dstNameSpace,
    NW_IDX N_FAR       *idxStruct);
```

Delphi Syntax

```
uses calwin32

Function NWNSGetMiscInfo
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   const path : pustr8;
   dstNameSpace : nuint8;
   Var idxStruct : NW_IDX
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to search.

path

(IN) Points to a valid NetWare path (pointing to a directory or a file).

dstNameSpace

(IN) Specifies the destination name space (see [Section 20.5, “Name Space Flag Values,”](#) on [page 595](#)).

idxStruct

(OUT) Points to NW_IDX.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89BF	INVALID_NAME_SPACE

Remarks

`dirHandle / path` should match `dstNameSpace`.

Both the `dirHandle` and `path` parameters must be in the default name space.

The default name space is the name space that matches the OS and the loaded name spaces on that volume. For example, Windows 95 on a volume with LONG name space will set LONG name space as the default name space.

NetWare uses NW_IDX as an index to quickly locate a directory entry (file or directory). NW_IDX is required as a parameter for other functions and should not be modified by the application.

NCP Calls

0x2222 87 06 Obtain File or Subdirectory Information

See Also

[NWGetDirectoryBase](#) (page 461)

NWNSRename

Renames an entry in the specified name space, given a path specifying the entry name

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWNSRename (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint8              namSpc,
    const nstr8 N_FAR  *oldName,
    nuint16             oldType,
    const nstr8 N_FAR  *newName,
    nuint8              renameFlag);
```

Delphi Syntax

```
uses calwin32;

Function NWNSRename
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   namSpc : nuint8;
   const oldName : pnstr8;
   oldType : nuint16;
   const newName : pnstr8;
   renameFlag : nuint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the parent directory.

namSpc

(IN) Specifies the name space of `oldName` (see [Section 20.5, “Name Space Flag Values,”](#) on [page 595](#)).

oldName

(IN) Points to the name of the directory or file to rename.

oldType

(IN) Specifies the type of `oldName`:

C Value	Delphi Value	Constant
0x8000	\$0800	NW_TYPE_FILE
0x0010	\$0010	NW_TYPE_SUBDIR

newName

(IN) Points to the new name (256 bytes maximum).

renameFlag

(IN) Specifies whether name conversion should be done; ignored for NetWare 3.11 and below:

C Value	Delphi Value	Constant
0x03	\$03	NW_NAME_CONVERT
0x04	\$04	NW_NO_NAME_CONVERT

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899E	INVALID_FILENAME

Remarks

A transaction file cannot be deleted or renamed.

`dirHandle` must point to the parent directory.

`oldName` and `newName` must be valid names containing only one component. `dirHandle` will specify the path.

The default operation for `NWNSRename` is to rename the file in all name spaces, report an error if renaming a file as itself, and do nothing with the file compatibility mode. When `NW_NAME_CONVERT` is passed in the `renameFlag` parameter, renaming the file to the same name will not report an error and compatibility mode will be set for that file. If `NW_NO_NAME_CONVERT` is passed in `renameFlag`, the new name is changed only in the specified name space. When renaming is done the shortening algorithm is used for the DOS and/or MAC name spaces when necessary.

AFP directory and file names (long names) contain 1-31 characters. A long name is a string preceded by one byte which specifies the length of the name. Long names can contain any ASCII character between 1 and 255 except the colon (:) but cannot be terminated by a NULL character (character 0).

The NetWare server automatically generates DOS-style file names (short names) for all AFP directories, as well as for created files and accessed files. The NetWare server maintains both the long name and the short name for each AFP directory and file.

For explanation of how long names are converted to DOS style names, see [“NetWare 4.x” on page 432](#) and [“NetWare 5.x and 6.x” on page 434](#).

NCP Calls

0x2222 23 17 Get File Server Information

0x2222 87 04 Rename Or Move A File Or Subdirectory

See Also

[NWGetLongName \(page 468\)](#)

NWNSRenameExt

Renames an entry in the specified name space, given a path specifying the entry name and using UTF-8 strings

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWNSRenameExt (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint8              namSpc,
    const nstr8 N_FAR  *oldName,
    nuint16             oldType,
    const nstr8 N_FAR  *newName,
    nuint8              renameFlag);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the parent directory. It cannot be zero.

namSpc

(IN) Specifies the name space of `oldName` (see [Section 20.5, “Name Space Flag Values,”](#) on [page 595](#)).

oldName

(IN) Points to the name of the directory or file to rename. The characters in the string must be UTF-8.

oldType

(IN) Specifies the type of `oldName`:

C Value	Constant
0x8000	NW_TYPE_FILE
0x0010	NW_TYPE_SUBDIR

newName

(IN) Points to the new name (256 characters maximum). The characters in the string must be UTF-8.

renameFlag

(IN) Specifies whether name conversion should be done:

C Value	Constant
0x03	NW_NAME_CONVERT
0x04	NW_NO_NAME_CONVERT

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899E	INVALID_FILENAME

Remarks

A transaction file cannot be deleted or renamed.

`dirHandle` must point to the parent directory.

`oldName` and `newName` must be valid names containing only one component. `dirHandle` will specify the path.

The default operation for `NWNSRenameExt` is to rename the file in all name spaces, report an error if renaming a file as itself, and do nothing with the file compatibility mode. When `NW_NAME_CONVERT` is passed in the `renameFlag` parameter, renaming the file to the same name will not report an error and compatibility mode will be set for that file. If `NW_NO_NAME_CONVERT` is passed in `renameFlag`, the new name is changed only in the

specified name space. When renaming is done the shortening algorithm is used for the DOS and/or MAC name spaces when necessary.

AFP directory and file names (long names) contain 1-31 characters. A long name is a string preceded by one byte which specifies the length of the name. Long names can contain any ASCII character between 1 and 255 except the colon (:) but cannot be terminated by a NULL character (character 0).

The NetWare server automatically generates DOS-style file names (short names) for all AFP directories, as well as for created files and accessed files. The NetWare server maintains both the long name and the short name for each AFP directory and file.

For explanation of how long names are converted to DOS style names, see [“NetWare 5.x and 6.x” on page 434](#).

NCP Calls

0x2222 23 17 Get File Server Information

0x2222 87 04 Rename Or Move A File Or Subdirectory

0x2222 89 04 Rename Or Move A File Or Subdirectory

See Also

[NWGetLongNameExt \(page 470\)](#)

NWOpenCreateNSEntry

Opens a file in the specified name space or creates and then opens a file if it does not already exist

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWOpenCreateNSEntry (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint8              namSpc,
    const pustr8        N_FAR path,
    NW_NS_OPENCREATE   N_FAR *NSOpenCreate,
    NWFH_HANDLE         N_FAR *fileHandle);
```

Delphi Syntax

```
uses calwin32;

Function NWOpenCreateNSEntry
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   namSpc : nuint8;
   const path : pustr8;
   Var NSOpenCreate : NW_NS_OPENCREATE;
   Var fileHandle : NWFH_HANDLE
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare connection handle.

dirHandle

(IN) Specifies the directory handle on which to open/create the specified file.

namSpc

(IN) Specifies the name space of `dirHandle/path` (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

path

(IN) Points to an absolute path (or relative if `dirHandle` is nonzero).

NSOpenCreate

(IN/OUT) Points to `NW_NS_OPENCREATE` containing information needed to create the entry on input. Points to `NW_NS_OPENCREATE` containing the results of a successful open/create upon output.

fileHandle

(OUT) Points to the `NWFILE_HANDLE`. When you are creating subdirectories, `fileHandle` returns zero.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8980	ERR_LOCK_FAIL
0x8981	NO_MORE_FILE_HANDLES
0x8982	NO_OPEN_PRIVILEGES
0x8994	NO_WRITE_PRIVILEGES_OR_READONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	SERVER_DOES_NOT_EXIST
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure

NCP Calls

0x2222 23 17 Get File Server Info
 0x2222 66 File Close
 0x2222 87 1 Open/Create Entry
 0x2222 87 30 Open/Create File or Subdirectory

See Also

[NWDeleteNSEntry \(page 455\)](#)

NWOpenCreateNSEntryExt

Opens a file in the specified name space or creates and then opens a file if it does not already exist. Path and file names must use UTF-8 characters.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWOpenCreateNSEntryExt (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint8              namSpc,
    const pnstr8        N_FAR path,
    NW_NS_OPENCREATE   N_FAR *NSOpenCreate,
    NWFH_HANDLE         N_FAR *fileHandle);
```

Parameters

conn

(IN) Specifies the NetWare connection handle.

dirHandle

(IN) Specifies the directory handle on which to open/create the specified file.

namSpc

(IN) Specifies the name space of dirHandle/path (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

path

(IN) Points to an absolute path (or relative if dirHandle is nonzero). The characters in the path string must be UTF-8.

NSOpenCreate

(IN/OUT) Points to NW_NS_OPENCREATE containing information needed to create the entry on input. Points to NW_NS_OPENCREATE containing the results of a successful open/create upon output.

fileHandle

(OUT) Points to the NWFILE_HANDLE. When you are creating subdirectories, fileHandle returns zero.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x88F0	UTF8_CONVERSION_FAILED
0x8980	ERR_LOCK_FAIL
0x8981	NO_MORE_FILE_HANDLES
0x8982	NO_OPEN_PRIVILEGES
0x8994	NO_WRITE_PRIVILEGES_OR_READONLY
0x8996	SERVER_OUT_OF_MEMORY
0x8998	SERVER_DOES_NOT_EXIST
0x899C	INVALID_PATH
0x89A1	DIRECTORY_IO_ERROR
0x89FD	BAD_STATION_NUMBER
0x89FF	Failure

NCP Calls

0x2222 23 17 Get File Server Info

0x2222 66 File Close

0x2222 87 1 Open/Create File or Subdirectory

0x2222 87 30 Open/Create File or Subdirectory

0x2222 89 1 Open/Create File or Subdirectory

0x2222 89 30 Open/Create File or Subdirectory

See Also

[NWDeleteNSEnterExt \(page 457\)](#), [NWOpenNSEnterExt \(page 519\)](#)

NWOpenDataStream

Opens a data stream associated with any supported name space on the server

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWOpenDataStream (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    const nstr8 N_FAR  *fileName,
    nuint16              dataStream,
    nuint16              attrs,
    nuint16              accessMode,
    pnuint32            NWHandle,
    NWFILE_HANDLE N_FAR *fileHandle);
```

Delphi Syntax

```
uses calwin32;

Function NWOpenDataStream
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   const fileName : pnstr8;
   dataStream : nuint16;
   attrs : nuint16;
   accessMode : nuint16;
   NWHandle : pnuint32;
   Var fileHandle : NWFILE_HANDLE
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory containing the file. This must be a DOS directory handle; if you want to read from a different namespace, use the `dataStream` parameter.

fileName

(IN) Points to the name of the file containing the data stream. It must be a DOS file name (and match `dirHandle`). For example, if you are opening a Macintosh file named "alongfilename," pass a DOS `dirHandle` and the DOS file name "ALONGFIL."

dataStream

(IN) Specifies the data stream number. To read the primary stream of any file in any namespace, pass 0. For example, to read the data fork of a file in the Macintosh namespace, open a DOS handle, and pass 0. To read the resource fork of a Macintosh handle, pass the DOS directory handle to `dirHandle`, and pass 1 as the data stream number.

0 NW_DS_DOS
 1 NW_DS_MAC
 2 NW_DS_FTAM

attrs

(IN) Specifies the attributes to use in searching for the file to open:

C Value	Delphi Value	Value Name
0x00	\$00	FA_NORMAL
0x01	\$01	FA_READ_ONLY
0x02	\$02	FA_HIDDEN
0x04	\$04	FA_SYSTEM
0x08	\$08	FA_EXECUTE_ONLY
0x10	\$10	FA_DIRECTORY
0x20	\$20	FA_NEEDS_ARCHIVED
0x80	\$80	FA_SHAREABLE

accessMode

(IN) Specifies the rights to use in opening the file (see [Section 20.1, "Access Right Values," on page 593](#)).

NWHandle

(OUT) Points to a 4-byte NetWare handle to `dataStream` (optional).

fileHandle

(OUT) Points to a file handle.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x8980	ERR_LOCK_FAIL
0x8982	NO_OPEN_PRIVILEGES
0x8990	NO_FILES_AFFECTED_READ_ONLY
0x89BE	INVALID_DATA_STREAM
0x89FF	NO_FILES_FOUND_ERROR

Remarks

NWOpenDataStream also obtains a NetWare file handle to a data stream.

If you pass a non-DOS namespace handle to `dirHandle`, NWOpenDataStream fails.

These constants identify trustee access rights for opening a directory with NWOpenDataStream.

C Value	Delphi Value	Value Name	Value Description
0x00	\$00	TA_NONE	Specifies no Reads or Writes are allowed.
0x01	\$01	TA_READ	Specifies file Reads are allowed.
0x02	\$02	TA_WRITE	Specifies file Writes are allowed.
0x08	\$08	TA_CREATE	Specifies files can be created.
0x10	\$10	TA_DELETE	Specifies files can be deleted.
0x20	\$20	TA_OWNERSHIP	Specifies subdirectories can be created or deleted and trustee rights granted or revoked.
0x40	\$40	TA_SEARCH	Specifies the directory can be searched.
0x80	\$80	TA_MODIFY	Specifies file attributes can be modified.
0xFB	\$FB	TA_ALL	Specifies the trustee has all the above rights to the directory.

NCP Calls

0x2222 22 49 Open Data Stream

0x2222 66 File Close

0x2222 87 06 Obtain File or Subdirectory Information

See Also

[NWAFPOpenFileFork](#) (Single and Intra-File Management), [NWConvertHandle](#) (page 168)

NWOpenNSEntry

Opens or creates a file or creates a subdirectory with a given owning name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWOpenNSEntry (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint8              nameSpc,
    nuint8              dataStream,
    const nstr8         N_FAR *path,
    NW_NS_OPEN          N_FAR *NSOpen,
    NWFILE_HANDLE      N_FAR *fileHandle);
```

Delphi Syntax

```
uses calwin32;

Function NWOpenNSEntry
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   namSpc : nuint8;
   dataStream : nuint8;
   const path : pnstr8;
   Var NSOpen : NW_NS_OPEN;
   Var fileHandle : NWFILE_HANDLE
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory in which to create the file.

namSpc

(IN) Specifies the name space for the file creation (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dataStream

(IN) Specifies the data stream number if the name space is Mac OS:

0 =Resource Fork

1=Data Fork

For DOS, always pass 0.

path

(IN) Points to the name to use in creating the file. Optionally contains a volume:path specification.

NSOpen

(IN/OUT) Points to NW_NS_OPENCREATE containing the information needed to open the entry. Results of a successful open are also returned in NW_NS_OPENCREATE.

fileHandle

(OUT) Points to the OS file handle; it returns zero if you are creating subdirectories.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

dirHandle can be zero if the path contains the complete path, including the volume name. (dirHandle/path should match namSpc.)

If you are creating a directory, pass NULL to fileHandle.

OC_MODE_ constants used in openCreateMode are listed below:

C Value	Delphi Value	Value Name
0x01	\$01	OC_MODE_OPEN
0x02	\$02	OC_MODE_TRUNCATE

C Value	Delphi Value	Value Name
0x02	\$02	OC_MODE_REPLACE
0x08	\$08	OC_MODE_CREATE

See [Section 20.8, “Search Attributes Values,” on page 597](#) for the possible values for the `searchAttributes` field.

See [Section 20.1, “Access Right Values,” on page 593](#) for the possible values for the `desiredAccessRights` field.

`OC_ACTION_` constants used in `openCreateAction` are listed below:

C Value	Delphi Value	Value Name
0x01	\$01	OC_ACTION_NONE
0x01	\$01	OC_ACTION_OPEN
0x02	\$02	OC_ACTION_CREATE
0x04	\$04	OC_ACTION_TRUNCATE
0x04	\$04	OC_ACTION_REPLACE

The file handle returned is appropriate for the platform the API is written for. This file handle may be used for access to the attribute value through standard file I/O with the handle. This includes closing the file as well as reading and writing to the file.

NOTE: When using this function to create a directory, the access rights field in the `NSOpen` structure is used to set the IRF on the created directory. Hence a value of `0xFF` should be used if an IRF of `[SRWCEMFA]` is required.

NCP Calls

0x2222 23 17 Get File Server Information
 0x2222 66 File Close
 0x2222 87 01 Open Create File Or Subdirectory
 0x2222 87 30 Open/Create File Or Subdirectory

See Also

[NWDeleteNSEntry \(page 455\)](#)

NWOpenNSEntryExt

Opens or creates a file or creates a subdirectory with a given owning name space and using UTF-8 strings.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWOpenNSEntryExt (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint8              namSpc,
    nuint8              dataStream,
    const nstr8         N_FAR *path,
    NW_NS_OPEN          N_FAR *NSOpen,
    NWFH_HANDLE         N_FAR *fileHandle);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory in which to create the file.

namSpc

(IN) Specifies the name space for the file creation (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dataStream

(IN) Specifies the data stream number if the name space is Mac OS:

0 =Resource Fork

1=Data Fork

For DOS, always pass 0.

path

(IN) Points to the name to use in creating the file. Optionally, it can point to an absolute path if `dirHandle` is zero. The characters in the string must be UTF-8.

NSOpen

(IN/OUT) Points to `NW_NS_OPEN` containing the information needed to open the entry. Results of a successful open are also returned in `NW_NS_OPENCREATE`.

fileHandle

(OUT) Points to the OS file handle; it returns zero if you are creating subdirectories. You can use this handle with the OS functions for reading, writing, and closing.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH

Remarks

`dirHandle` can be zero if the path contains the complete path, including the volume name. (`dirHandle / path` should match `namSpc`.)

If you are creating a directory, pass `NULL` to `fileHandle`.

`OC_MODE_` constants used in `openCreateMode` are listed below:

C Value	Value Name
0x01	OC_MODE_OPEN
0x02	OC_MODE_TRUNCATE
0x02	OC_MODE_REPLACE
0x08	OC_MODE_CREATE

See [Section 20.8, “Search Attributes Values,” on page 597](#) for the possible values for the `searchAttributes` field.

See [Section 20.1, “Access Right Values,” on page 593](#) for the possible values for the `desiredAccessRights` field.

OC_ACTION_ constants used in openCreateAction are listed below:

C Value	Value Name
0x01	OC_ACTION_NONE
0x01	OC_ACTION_OPEN
0x02	OC_ACTION_CREATE
0x04	OC_ACTION_TRUNCATE
0x04	OC_ACTION_REPLACE

The file handle returned is appropriate for the platform the API is written for. This file handle may be used for access to the attribute value through standard file I/O with the handle. This includes closing the file as well as reading and writing to the file.

NCP Calls

0x2222 23 17 Get File Server Information
0x2222 66 File Close
0x2222 87 01 Open/Create File Or Subdirectory
0x2222 87 30 Open/Create File Or Subdirectory
0x2222 89 01 Open/Create File Or Subdirectory
0x2222 89 30 Open/Create File Or Subdirectory

See Also

[NWDeleteNSEntryExt \(page 457\)](#), [NWOpenCreateNSEntryExt \(page 510\)](#)

NWQueryNSInfoFormat

Returns the NW_NS_INFO structure to be used in getting and setting name space information

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>

int NWQueryNSInfoFormat (
    BYTE          nameSpace,
    BYTE          volNum,
    NW_NS_INFO    *nsInfo);
```

Parameters

nameSpace

(IN) Specifies the name space to return information for (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

volNum

(IN) Specifies the volume number to return information for.

nsInfo

(OUT) Points to an NW_NS_INFO structure.

Return Values

ESuccess or NetWare errors

Remarks

The nsInfo parameter points to an NW_NS_INFO structure. This structure is defined in nwnspace.h as follows:

```
typedef struct
{
    LONG    nsInfoBitMask;
    LONG    fixedBitMask;
    LONG    reservedBitMask;
    LONG    extendedBitMask;
    WORD    fixedBitsDefined;
    WORD    reservedBitsDefined;
```

```
WORD    extendedBitsDefined;
LONG    fieldsLenTable[32];
BYTE    hugeStateInfo[16];
LONG    hugeDataLength;
}    NW_NS_INFO;
```

See Also

[NWGetNSInfo \(NLM\) \(page 483\)](#), [NWSetNSInfo \(page 557\)](#)

NWReadExtendedNSInfo

Reads the extended (huge) name space information for the specified name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWReadExtendedNSInfo (
    NWCONN_HANDLE      conn,
    const NW_IDX N_FAR *idxStruct,
    NW_NS_INFO N_FAR   *NSInfo,
    puint8              data);
```

Delphi Syntax

```
uses calwin32;

Function NWReadExtendedNSInfo
  (conn : NWCONN_HANDLE;
   Var idxStruct : NW_IDX;
   Var NSInfo : NW_NS_INFO;
   data : puint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

idxStruct

(IN) Points to NW_IDX returned from NWNSGetMiscInfo.

NSInfo

(IN) Points to NW_NS_INFO returned from NWGetNSInfo.

data

(OUT) Points to a buffer containing the data from the name space.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION

Remarks

If `extendedBitMask` is set in `NW_NS_INFO`, `NWReadExtendedNSInfo` should be used to read the extended information. `extendedBitMask` contains a Read-only information field that should be preserved. The application must not manipulate `extendedBitMask` ; it must not be zero.

`dstNameSpace` and `dstDirBase` of `NW_IDX` are used to determine the target name space of `NWReadExtendedNSInfo`.

NCP Calls

0x2222 87 26 Get Huge NS Information

See Also

[NWGetDirectoryBase \(page 461\)](#), [NWGetNSInfo \(page 481\)](#), [NWNSGetMiscInfo \(page 500\)](#), [NWWriteExtendedNSInfo \(page 559\)](#)

NWReadNSInfo

Reads name space information from the designated name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWReadNSInfo (
    NWCONN_HANDLE      conn,
    const NW_IDX N_FAR  *idxStruct,
    const NW_NS_INFO N_FAR *NSInfo,
    puint8              data);
```

Delphi Syntax

```
uses calwin32;

Function NWReadNSInfo
  (conn : NWCONN_HANDLE;
   Var idxStruct : NW_IDX;
   Var NSInfo : NW_NS_INFO;
   data : puint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

idxStruct

(IN) Points to NW_IDX returned from NWNSGetMiscInfo.

NSInfo

(IN) Points to NW_NS_INFO returned from NWGetNSInfo.

data

(OUT) Points to a 512-byte buffer receiving data from the name space.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION

Remarks

NSInfoBitMask bit definitions follow:

C Value	Delphi Value	Constant
0x0002L	\$0002	DM_ATTRIBUTES
0x0004L	\$0004	DM_CREATE_DATE
0x0008L	\$0008	DM_CREATE_TIME
0x0010L	\$0010	DM_CREATOR_ID
0x0020L	\$0020	DM_ARCHIVE_DATE
0x0040L	\$0040	DM_ARCHIVE_TIME
0x0080L	\$0080	DM_ARCHIVER_ID
0x0100L	\$0100	DM_MODIFY_DATE
0x0200L	\$0200	DM_MODIFY_TIME
0x0400L	\$0400	DM_MODIFIER_ID
0x0800L	\$0800	DM_LAST_ACCESS_DATE
0x1000L	\$1000	DM_INHERITED_RIGHTS_MASK
0x2000L	\$2000	DM_MAXIMUM_SPACE

NCP Calls

0x2222 87 19 Get NS Information

See Also

[NWGetNSEntryInfo \(page 474\)](#), [NWWriteNSInfo \(page 561\)](#)

NWReadNSInfoExt

Reads name space information from the designated name space, using UTF-8 strings.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWReadNSInfoExt (
    NWCONN_HANDLE      conn,
    const NW_IDX        N_FAR *idxStruct,
    const NW_NS_INFO   N_FAR *NSInfo,
    puint8              data);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

idxStruct

(IN) Points to NW_IDX returned from [NWGetDirectoryBaseExt \(page 464\)](#).

NSInfo

(IN) Points to NW_NS_INFO returned from [NWGetNSInfo \(page 481\)](#).

data

(OUT) Points to a 1024-byte buffer receiving data from the name space.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED

0x890A	NLM_INVALID_CONNECTION
--------	------------------------

Remarks

NSInfoBitMask bit definitions follow:

C Value	Constant
0x0002L	DM_ATTRIBUTES
0x0004L	DM_CREATE_DATE
0x0008L	DM_CREATE_TIME
0x0010L	DM_CREATOR_ID
0x0020L	DM_ARCHIVE_DATE
0x0040L	DM_ARCHIVE_TIME
0x0080L	DM_ARCHIVER_ID
0x0100L	DM_MODIFY_DATE
0x0200L	DM_MODIFY_TIME
0x0400L	DM_MODIFIER_ID
0x0800L	DM_LAST_ACCESS_DATE
0x1000L	DM_INHERITED_RIGHTS_MASK
0x2000L	DM_MAXIMUM_SPACE

NCP Calls

0x2222 87 19 Get NS Information

0x2222 89 19 Get NS Information

See Also

[NWGetNSEntryInfoExt \(page 477\)](#), [NWGetNSInfo \(page 481\)](#), [NWWriteNSInfoExt \(page 563\)](#)

NWScanNSDirectoryForTrustees

Scans a directory for trustees using the specified path and directory handle under a specified name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWScanNSDirectoryForTrustees (
    NWCONN_HANDLE          conn,
    nuint8                  namSpc,
    nuint8                  dirHandle,
    const nstr8 N_FAR      *srchPath,
    pnuint32                iterHandle,
    pustr8                  dirName,
    pnuint32                dirDateTime,
    pnuint32                ownerID,
    TRUSTEE_INFO N_FAR     *trusteeList);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

namSpc

(IN) Specifies the name space of the dirHandle/srchPath combination (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dirHandle

(IN) Specifies the NetWare directory handle for the directory being scanned (0 if the srchPath parameter points to the complete path, including the volume name)

srchPath

(IN) Points to an absolute directory path (or a path relative to the directory handle) and a search pattern

iterHandle

(IN/OUT) Points to the sequence number to be used for subsequent calls (0 initially)

dirName

(OUT) Points to the directory name found (optional, up to 256 bytes)

dirDateTime

(OUT) Points to the creation date and time of the directory (optional)

ownerID

(OUT) Points to the object ID of the directory owner (optional)

trusteeList

(OUT) Points to an array of 20 TRUSTEE_INFO structures

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	NO_MORE_TRUSTEES
	INVALID_PATH

Remarks

Directories can have any number of objects as trustees. The directory trustees are stored and retrieved in groups on the server. To obtain a complete list, use the `iterHandle` parameter.

`NWScanNSDirectoryForTrustees` increments the value referenced by the `iterHandle` parameter to the next appropriate value. For subsequent calls, pass in the new value of the `iterHandle` parameter.

Trustees are returned in groups of 20 `TRUSTEE_INFO` structures. Due to subtle differences in operation, trustees may remain after an iteration, even though not all 20 positions are filled. If a position is not filled, the `objectID` field of `TRUSTEE_INFO` has a value of 0L.

`NWScanNSDirectoryForTrustees` should be called until `iterHandle` is -1 or it returns 0x899C (`NO_MORE_TRUSTEES`). Because 0x899C also means `INVALID_PATH`, ensure the `dirHandle/pbstrSrchPath` parameter combination is correct.

NULL can be substituted for all optional items. However, all parameter positions must be filled.

NCP Calls

0x2222 87 05 Scan File Or Subdirectory For Trustees
0x2222 87 06 Obtain File or Subdirectory Information

See Also

[NWAddTrustee \(page 153\)](#), [NWAddTrusteeToDirectory \(page 158\)](#), [NWAddTrusteeToNSDirectory \(page 448\)](#), [NWDeleteTrustee \(page 177\)](#), [NWDeleteTrusteeFromDirectory \(page 181\)](#), [NWDeleteTrusteeFromNSDirectory \(page 459\)](#), [NWScanDirectoryForTrustees2 \(page 262\)](#)

NWScanNSEntryInfo

Obtains directory entry information using a specific name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWScanNSEntryInfo (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint8              namSpc,
    nuint16             attrs,
    SEARCH_SEQUENCE N_FAR *sequence,
    const nstr8 N_FAR  *srchPattern,
    nuint32             retInfoMask,
    NW_ENTRY_INFO N_FAR *entryInfo);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWScanNSEntryInfo
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   namSpc : nuint8;
   attrs : nuint16;
   Var sequence : SEARCH_SEQUENCE;
   const searchPattern : pnstr8;
   retInfoMask : nuint32;
   Var entryInfo : NW_ENTRY_INFO
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to scan. Must point to the parent directory.

namSpc

(IN) Specifies the name space of `dirHandle` (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

attr

(IN) Specifies the attributes to be used for the scan (see [Section 20.8, “Search Attributes Values,” on page 597](#)).

sequence

(IN/OUT) Points to `SEARCH_SEQUENCE`.

srchPattern

(IN) Points to the name of the entry for which to scan (wildcards are allowed).

retInfoMask

(IN) Specifies the information to return (see [Section 20.6, “Basic Return Mask Values,” on page 595](#) and don't use the Extended Return Mask Values).

entryInfo

(OUT) Points to `NW_ENTRY_INFO`.

Return Values

These are common return values; see [Return Values \(*Return Values for C*\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x890A	NLM_INVALID_CONNECTION

Remarks

`NWScanNSEntryInfo` can be used iteratively with wild cards. On the first call, `searchDirNumber` in the `SEARCH_SEQUENCE` structure should be set to -1. After that, the server manages the information.

`retInfoMask` is used to determine which fields of `NW_ENTRY_INFO` to return. `nameLength` and `entryName` are always returned in `NWScanNSEntryInfo`.

To request information from a server, a client sets the appropriate bit or bits of `retInfoMask` and sends a request packet to the server.

NCP Calls

0x2222 87 02 Initialize Search

0x2222 87 03 Search For File Or Subdirectory

See Also

[NWGetNSEntryInfo \(page 474\)](#)

NWScanNSEntryInfoExt

Obtains directory entry information, using a specific name space and UTF-8 strings.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWScanNSEntryInfoExt (
    NWCONN_HANDLE          conn,
    nuInt8                  dirHandle,
    nuInt8                  namSpc,
    nuInt16                 attrs,
    SEARCH_SEQUENCE        N_FAR *sequence,
    const nstr8             N_FAR *srchPattern,
    nuInt32                 retInfoMask,
    NW_ENTRY_INFO_EXT      N_FAR *entryInfo);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to scan. Must point to the parent directory.

namSpc

(IN) Specifies the name space of `dirHandle` (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

attr

(IN) Specifies the attributes to be used for the scan (see [Section 20.8, “Search Attributes Values,” on page 597](#)).

sequence

(IN/OUT) Points to `SEARCH_SEQUENCE`.

srchPattern

(IN) Points to the name of the entry for which to scan (wildcards are allowed).

retInfoMask

(IN) Specifies the information to return (see [Section 20.6, “Basic Return Mask Values,”](#) on [page 595](#) and don't use the Extended Return Mask Values).

entryInfo

(OUT) Points to NW_ENTRY_INFO_EXT.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x890A	NLM_INVALID_CONNECTION

Remarks

NWScanNSEntryInfoExt can be used iteratively with wild cards. On the first call, `searchDirNumber` in the SEARCH_SEQUENCE structure should be set to -1. After that, the server manages the information.

`retInfoMask` is used to determine which fields of NW_ENTRY_INFO to return. `nameLength` and `entryName` are always returned in NWScanNSEntryInfoExt.

To request information from a server, a client sets the appropriate bit or bits of `retInfoMask` and sends a request packet to the server.

NCP Calls

0x2222 87 02 Initialize Search
 0x2222 87 03 Search For File Or Subdirectory
 0x2222 89 02 Initialize Search
 0x2222 89 03 Search For File Or Subdirectory

See Also

[NWGetNSEntryInfoExt \(page 477\)](#)

NWScanNSEntryInfo2

Obtains directory entry information, returning more information and using network bandwidth more efficiently than the NWScanNSEntryInfo function.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY (NWCCODE) NWScanNSEntryInfo2 (
    NWCONN_HANDLE          conn,
    nuint8                  dirHandle,
    nuint8                  namSpc,
    nuintl6                 attrs,
    SEARCH_SEQUENCE N_FAR  *sequence,
    const nstr8 N_FAR      *srchPattern,
    nuint32                 retInfoMask,
    NW_ENTRY_INFO2 N_FAR  *entryInfo);
```

Delphi Syntax

```
Function NWScanNSEntryInfo2 (
    conn : NWCONN_HANDLE;
    dirHandle: nuint8;
    namSpc: nuint8;
    attrs : nuintl6;
    Var sequence : SEARCH_SEQUENCE;
    const srchPattern : pustr8;
    retInfoMask : nuint32;
    Var entryInfo : NW_ENTRY_INFO2
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to scan (must be valid and cannot be zero).

namSpc

(IN) Specifies the name space of `dirHandle` (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

attr

(IN) Specifies the attributes to be used for the scan (see [Section 20.8, “Search Attributes Values,”](#) on page 597).

sequence

(IN/OUT) Points to `SEARCH_SEQUENCE`.

srchPattern

(IN) Points to the name of the entry for which to scan (wildcards are allowed).

retInfoMask

(IN) Specifies the information to return (see [Section 20.6, “Basic Return Mask Values,”](#) on page 595 and [Section 20.7, “Extended Return Mask Values,”](#) on page 596).

entryInfo

(OUT) Points to the `NW_ENTRY_INFO2` structure.

Return Values

These are common return values; see [Return Values \(*Return Values for C*\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8813	INVALID_DIR_HANDLE
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x890A	NLM_INVALID_CONNECTION
0x89FF	NO_FILES_FOUND

Remarks

`NWScanNEntryInfo2` can be used iteratively with wildcards. On the first iteration, set `searchDirNumber` in the `SEARCH_SEQUENCE` structure to -1. After that, the server manages the information.

The `retInfoMask` parameter is used to determine which fields of `NW_ENTRY_INFO2` to return; `nameLength` and `entryName` are always returned in `NWScanNEntryInfo2`.

To request information from a server, a client sets the appropriate bit or bits of `retInfoMask` and sends a request packet to the server.

NCP Calls

0x2222 87 02 Initialize Search

0x2222 87 03 Search For File Or Subdirectory

See Also

[NWGetNSEntryInfo \(page 474\)](#), [NWScanNSEntryInfo \(page 533\)](#)

NWScanNSEntryInfoSet

Scans a set of directory and file entry information by using a specific name space.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT*, Windows* 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE NWScanNSEntryInfoSet (
    NWCONN_HANDLE          conn,
    NWDIR_HANDLE           dirHandle,
    nuint8                  buNameSpace,
    nuint16                 suAttr,
    SEARCH_SEQUENCE N_FAR  *pIterHnd,
    const nstr8 N_FAR      *pbstrSrchPattern,
    nuint32                 luRetMask,
    pnuint8                 pbuMoreEntriesFlag,
    punint16                psuNumReturned,
    nuint16                 suNumItems,
    NW_ENTRY_INFO N_FAR    *pEntryInfo);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle associated with the directory to be scanned (must be the parent directory handle).

buNameSpace

(IN) Specifies the name space of `dirHandle` (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

suAttr

(IN) Specifies the attributes to be used for the scan (see [Section 20.8, “Search Attributes Values,” on page 597](#)).

pIterHnd

(IN/OUT) Points to SEARCH_SEQUENCE.

pbstrSrchPattern

(IN) Points to the name of the entry for which to scan (wildcards are allowed).

luRetMask

(IN) Specifies which information is to be returned in the array pointed to by pEntryInfo (see [Section 20.6, “Basic Return Mask Values,”](#) on page 595-*this parameter cannot take Extended Return Mask Values*).

pbuMoreEntriesFlag

(OUT) Points to a flag indicating whether more entries are available:

0xFF More entries are available

0 No more entries are available

psuNumReturned

(OUT) Points to a value indicating how many NW_ENTRY_INFO structures were actually returned in the pEntryInfo array.

suNumItems

(IN) Specifies the size of the array pointed to by pEntryInfo.

pEntryInfo

(OUT) Points to an array of NW_ENTRY_INFO structures.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x890A	NLM_INVALID_CONNECTION

Remarks

NWScanNSEntryInfoSet is a version of NWScanNSEntryInfo that has been enhanced to return a list of entry information.

For the first request, the searchDirNumber field in SEARCH_SEQUENCE should be set to -1. Thereafter, the server manages the information; users should never directly change the value.

NCP Calls

0x2222 87 20 Search for File or SubDirectory Set

See Also

[NWGetNSEntryInfo \(page 474\)](#), [NWGetNSInfo \(page 481\)](#), [NWSetNSInfo \(page 557\)](#)

NWSetHugeNSInfo

Sets extended (huge) NS information for the entry specified by `volNum`, `nameSpace`, and `dirBase`

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>

int NWSetHugeNSInfo (
    BYTE    volNum,
    BYTE    nameSpace,
    LONG    dirBase,
    LONG    hugeInfoMask,
    BYTE    *hugeStateInfo,
    LONG    *hugeDataLen,
    BYTE    *hugeData,
    BYTE    *nextHugeStateInfo,
    LONG    *hugeDataUsed);
```

Parameters

volNum

(IN) Specifies the volume number for which to set huge NS information.

nameSpace

(IN) Specifies the name space for which to set huge information (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dirBase

(IN) Specifies the directory base (or number) for the entry for which to set information.

hugeInfoMask

(IN) Specifies the bit map that indicates which types of information is being set.

hugeStateInfo

(IN) Points to the information that helps the name space transfer the data across the wire. The `hugeStateInfo` is information that was returned by a previous call to `NWGetHugeNSInfo`.

hugeDataLen

(IN) Points to the length of the huge data to be set.

hugeData

(IN) Points to the data to be set as specified in the hugeInfoMask.

nextHugeStateInfo

(OUT) Points to the huge state information that should be passed in on the next call to this function should all the information not fit in one packet.

hugeDataUsed

(OUT) Points to the number of bytes that were actually set by the name space.

Return Values

ESuccess or NetWare errors

Remarks

This function sets extended NS information for an entry in the specified name space.

See Also

[NWGetDirBaseFromPath \(page 610\)](#), [NWGetHugeNSInfo \(page 466\)](#), [NWQueryNSInfoFormat \(page 522\)](#)

NWSetLongName

Renames an entry in the specified name space, given a path specifying the entry name

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWSetLongName (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    nuint8              namSpc,
    const nstr8 N_FAR  *dstPath,
    nuint16             dstType,
    const nstr8 N_FAR  *longName);
```

Delphi Syntax

```
uses calwin32;

Function NWSetLongName
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   namSpc : nuint8;
   dstPath : pnstr8;
   dstType : nuint16;
   longName : pnstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the parent directory.

namSpc

(IN) Specifies the name space of `dstPath` (see [Section 20.5, “Name Space Flag Values,”](#) on [page 595](#)).

dstPath

(IN) Points to the name of the directory or file to rename.

dstType

(IN) Specifies the directory or file type that `dstPath` points to.

longName

(IN) Points to the new name (256 bytes maximum).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x899E	INVALID_FILENAME

Remarks

`dirHandle` must point to the parent directory.

`dstPath` and `longName` must be valid names containing only one component. `dirHandle` will specify the path where the one component is located.

`dstType` can take on the following values:

C Value	Delphi Value	Value Name
0x8000	\$0800	NW_TYPE_FILE
0x0010	\$0010	NW_TYPE_SUBDIR

Resetting a filename in one name space resets the name in all name spaces. The shortening algorithm is used for the DOS and/or Macintosh name spaces, if appropriate.

AFP directory and file names contain from 1 to 31 characters and consist of a Delphi string preceded by one byte which specifies the length of the name. AFP names can contain any ASCII character between 1 and 255 except the colon (:) but cannot be terminated by a NULL character (character 0). NetWare servers automatically generate DOS-style file names (short names) for all AFP directories,

as well as for created files and accessed files. NetWare servers maintain both the AFP name and the short name for each AFP directory and file.

For explanation of how long names are converted to DOS style names, see [“NetWare 4.x” on page 432](#) and [“NetWare 5.x and 6.x” on page 434](#).

NCP Calls

0x2222 23 17 Get File Server Information

0x2222 87 04 Rename Or Move A File Or Subdirectory

See Also

[NWGetLongName \(page 468\)](#)

NWSetNameSpaceEntryName

Sets the name of a file or directory in the specified name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>

int NWSetNameSpaceEntryName (
    BYTE    *path,
    LONG    nameSpace,
    BYTE    *nameSpaceEntryName);
```

Parameters

path

(IN) Points to the path of the file system entry to set a name space entry name for.

nameSpace

(IN) Specifies the name space to set the file or directory name for (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

nameSpaceEntryName

(IN) Points to an ASCII string that specifies the new file or directory name in the specified name space.

Return Values

ESuccess or NetWare errors

Remarks

This function sets the file system entry's name in the specified name space only. The naming change is not reflected in the other name space entries.

See Also

[NWSetNameSpaceEntryName \(page 549\)](#)

Example

See the example for [NWGetNamespaceEntryName](#) (page 472).

NWSetNSEntryDOSInfo

Modifies information in one name space using a path from another name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWSetNSEntryDOSInfo (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    const nstr8 N_FAR  *path,
    nuint8              namSpc,
    nuint16             searchAttrs,
    nuint32             modifyDOSMask,
    MODIFY_DOS_INFO N_FAR *dosInfo);
```

Delphi Syntax

```
uses calwin32;

Function NWSetNSEntryDOSInfo
  (conn : NWCONN_HANDLE;
   dirHandle : nuint8;
   path : pnstr8;
   namSpc : nuint8;
   searchAttrs : nuint16;
   modifyDOSMask : nuint32;
   Var dosInfo : MODIFY_DOS_INFO
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the parent directory.

path

(IN) Points to the path.

namSpc

(IN) Specifies the name space of `dirHandle` and `path` (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

searchAttrs

(IN) Specifies the search attributes to use.

modifyDOSMask

(IN) Specifies the information to set.

dosInfo

(IN) Points to `MODIFY_DOS_INFO` containing the information specified by `luModifyDOSMask`.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x89FF	NO_FILES_FOUND_ERROR

Remarks

`suSrchAttr` can have the following values:

C Value	Delphi Value	Value Name
0x0002	\$0002	SA_HIDDEN
0x0004	\$0004	SA_SYSTEM
0x0010	\$0010	SA_SUBDIR_ONLY
0x8000	\$8000	SA_SUBDIR_FILES

`luModifyDOSMask` can have the following values:

C Value	Delphi Value	Value Name
0x0002L	\$0002	DM_ATTRIBUTES
0x0004L	\$0004	DM_CREATE_DATE
0x0008L	\$0008	DM_CREATE_TIME
0x0010L	\$0010	DM_CREATOR_ID
0x0020L	\$0020	DM_ARCHIVE_DATE
0x0040L	\$0040	DM_ARCHIVE_TIME

C Value	Delphi Value	Value Name
0x0080L	\$0080	DM_ARCHIVER_ID
0x0100L	\$0100	DM_MODIFY_DATE
0x0200L	\$0200	DM_MODIFY_TIME
0x0400L	\$0400	DM_MODIFIER_ID; cannot be set for subdirectories
0x0800L	\$0800	DM_LAST_ACCESS_DATE; cannot be set for subdirectories
0x1000L	\$1000	DM_INHERITED_RIGHTS_MASK
0x2000L	\$2000	DM_MAXIMUM_SPACE

DM_MODIFIER_ID and DM_LAST_ACCESS_DATE cannot be used when the `suSrchAttr` parameter contains SA_SUBDIR_ONLY. The server masks off DM_MODIFIER_ID and DM_LAST_ACCESS_DATE on subdirectories. If the resultant mask is 0x0000, the server will return NO_FILES_FOUND_ERROR indicating DM_MODIFIER_ID and DM_LAST_ACCESS_DATE were not set. If the resultant mask still contains a return value other than SUCCESSFUL, NWSetNSEntryDOSInfo will set the remaining bits and return SUCCESSFUL even though DM_MODIFIER_ID and DM_LAST_ACCESS_DATE were not set.

NCP Calls

0x2222 87 07 Modify File or Subdirectory DOS Information

NWSetNSEntryDOSInfoExt

Modifies information in one name space using a path from another name space and UTF-8 strings

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWSetNSEntryDOSInfoExt (
    NWCONN_HANDLE      conn,
    nuint8              dirHandle,
    const nstr8         N_FAR *path,
    nuint8              namSpc,
    nuint16             searchAttrs,
    nuint32             modifyDOSMask,
    MODIFY_DOS_INFO N_FAR *dosInfo);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle of the parent directory.

path

(IN) Points to the path. The characters in the string must be UTF-8.

namSpc

(IN) Specifies the name space of `dirHandle` and `path` (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

searchAttrs

(IN) Specifies the search attributes to use (see Remarks for values).

modifyDOSMask

(IN) Specifies the information to set (see Remarks for values).

dosInfo

(IN) Points to MODIFY_DOS_INFO containing the information specified by `luModifyDOSMask`.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x88F0	UTF8_CONVERSION_FAILED
0x89FF	NO_FILES_FOUND_ERROR

Remarks

`suSrchAttr` can have the following values:

C Value	Value Name
0x0002	SA_HIDDEN
0x0004	SA_SYSTEM
0x0010	SA_SUBDIR_ONLY
0x8000	SA_SUBDIR_FILES

`luModifyDOSMask` can have the following values:

C Value	Value Name
0x0002L	DM_ATTRIBUTES
0x0004L	DM_CREATE_DATE
0x0008L	DM_CREATE_TIME
0x0010L	DM_CREATOR_ID
0x0020L	DM_ARCHIVE_DATE
0x0040L	DM_ARCHIVE_TIME
0x0080L	DM_ARCHIVER_ID
0x0100L	DM_MODIFY_DATE
0x0200L	DM_MODIFY_TIME
0x0400L	DM_MODIFIER_ID; cannot be set for subdirectories
0x0800L	DM_LAST_ACCESS_DATE; cannot be set for subdirectories

C Value	Value Name
0x1000L	DM_INHERITED_RIGHTS_MASK
0x2000L	DM_MAXIMUM_SPACE

DM_MODIFIER_ID and DM_LAST_ACCESS_DATE cannot be used when the `suSrchAttr` parameter contains SA_SUBDIR_ONLY. The server masks off DM_MODIFIER_ID and DM_LAST_ACCESS_DATE on subdirectories. If the resultant mask is 0x0000, the server will return NO_FILES_FOUND_ERROR indicating DM_MODIFIER_ID and DM_LAST_ACCESS_DATE were not set. If the resultant mask still contains a return value other than SUCCESSFUL, NWSetNSEntryDOSInfoExt will set the remaining bits and return SUCCESSFUL even though DM_MODIFIER_ID and DM_LAST_ACCESS_DATE were not set.

NCP Calls

0x2222 87 07 Modify File or Subdirectory DOS Information

0x2222 89 07 Modify File or Subdirectory DOS Information

NWSetNSInfo

Sets specific NS information for a directory entry specified by `volNum`, `nameSpace`, and `dirBase`

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

Service: Name Space

Syntax

```
#include <nwnspace.h>
```

```
int NWSetNSInfo (
    BYTE    volNum,
    BYTE    srcNameSpace,
    BYTE    dstNameSpace,
    LONG    dirBase,
    LONG    nsInfoMask,
    LONG    nsSpecificInfoLen,
    BYTE    *nsSpecificInfo);
```

Parameters

volNum

(IN) Specifies the volume number for which information is being set.

srcNameSpace

(IN) Specifies the name space that corresponds with the `dirBase` being passed (see [Section 20.5, “Name Space Flag Values,” on page 595](#)). The name space currently being worked with is the default.

dstNameSpace

(IN) Specifies the name space to which information is being set (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

dirBase

(IN) Specifies the directory base (or number) for the entry on which information is being set.

nsInfoMask

(IN) Specifies the bit map that indicates which types of information the user is setting in the data parameter.

nsSpecificInfoLen

(IN) Specifies the length of the data being set.

nsSpecificInfo

(IN) Points to that is being set as indicated in the `nsInfoMask`.

Return Values

ESuccess or NetWare errors

Remarks

If the current name space is NFS, a value of 2 (for NFS) would be passed as `srcNameSpace`. If, however, the returned information should be in another format, for example LONG, a value of 4 would be passed as the `dstNameSpace`.

See [“DOS Name Space Bit Mask” on page 431](#).

See Also

[NWGetDirBaseFromPath \(page 610\)](#), [NWGetNSInfo \(NLM\) \(page 483\)](#), [NWQueryNSInfoFormat \(page 522\)](#)

NWWriteExtendedNSInfo

Writes the extended (huge) name space information for the specified name space

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWWriteExtendedNSInfo (
    NWCONN_HANDLE      conn,
    const NW_IDX N_FAR *idxStruct,
    NW_NS_INFO N_FAR *NSInfo,
    const nstr8 N_FAR *data);
```

Delphi Syntax

```
uses calwin32

Function NWWriteExtendedNSInfo
  (conn : NWCONN_HANDLE;
   Var idxStruct : NW_IDX;
   Var NSInfo : NW_NS_INFO;
   data : puint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

idxStruct

(IN) Points to NW_IDX returned by NWNSGetMiscInfo.

NSInfo

(IN) Points to NW_NS_INFO returned by NWGetNSInfo.

data

(IN) Points to a buffer containing the data to be written to the name space.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x898C	NO_MODIFY_PRIVILEGES

Remarks

`dstNameSpace` and `dstDirBase` in `NW_IDX` are used to determine what entry to use for the Write.

`extendedBitMask` in `NW_NS_INFO` is a read-only information field that should be preserved from `NWReadExtendedNSInfo`.

NCP Calls

0x2222 87 27 Set Huge NS Information

See Also

[NWGetDirectoryBase](#) (page 461), [NWGetNSInfo](#) (page 481), [NWNSGetMiscInfo](#) (page 500), [NWReadExtendedNSInfo](#) (page 524), [NWWriteExtendedNSInfo](#) (page 559)

NWWriteNSInfo

Sets the specific name space information

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>
```

```
NWCCODE N_API NWWriteNSInfo (
    NWCONN_HANDLE      conn,
    const NW_IDX N_FAR  *idxStruct,
    const NW_NS_INFO N_FAR *NSInfo,
    const nstr8 N_FAR   *data);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWWriteNSInfo
  (conn : NWCONN_HANDLE;
   Var idxStruct : NW_IDX;
   Var NSInfo : NW_NS_INFO;
   data : puint8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

idxStruct

(IN) Points to NW_IDX returned by NWNSGetMiscInfo.

NSInfo

(IN) Points to NW_NS_INFO returned by NWGetNSInfo.

data

(IN) Points to a 512-byte buffer containing the data to be written to the name space.

Return Values

These are common return values; see [Return Values](#) (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION

Remarks

For name spaces other than DOS, NWWriteNSInfo is passed to the appropriate name space NLM on the server. For the DOS name space, the server processes the request.

The actual format of the data is determined by the NLM on the server. Unless format for the data on the server is known, NWWriteNSInfo should not be used.

Avoid setting the first field of the name space information. This is generally the name and is intended to be read-only. To rename a file, call NWSetLongName.

NCP Calls

0x2222 87 25 Set NS Information

See Also

[NWGetDirectoryBase](#) (page 461), [NWGetNSInfo](#) (page 481), [NWNSGetMiscInfo](#) (page 500), [NWReadNSInfo](#) (page 526)

NWWriteNSInfoExt

Sets the specific name space information, using UTF-8 strings.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 6.5 SP2 or later

Platform: NLM, Windows 2000, Windows XP

Client: 4.90 SP2 or later

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwnamspc.h>
or
#include <nwcalls.h>

NWCCODE N_API NWWriteNSInfoExt (
    NWCONN_HANDLE      conn,
    const NW_IDX        N_FAR *idxStruct,
    const NW_NS_INFO    N_FAR *NSInfo,
    const nstr8         N_FAR *data);
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

idxStruct

(IN) Points to NW_IDX returned from [NWGetDirectoryBaseExt \(page 464\)](#).

NSInfo

(IN) Points to NW_NS_INFO returned by [NWGetNSInfo \(page 481\)](#).

data

(IN) Points to a 1024-byte buffer containing the data to be written to the name space.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x88F0	UTF8_CONVERSION_FAILED

Remarks

For name spaces other than DOS, NWWriteNSInfoEXT is passed to the appropriate name space NLM on the server. For the DOS name space, the server processes the request.

The actual format of the data is determined by the NLM on the server. Unless format for the data on the server is known, NWWriteNSInfoEXT should not be used.

Avoid setting the first field of the name space information. This is generally the name and is intended to be read-only. To rename a file, call NWNSRenameExt.

NCP Calls

0x2222 87 25 Set NS Information

0x2222 89 25 Enhanced Set NS Information

See Also

[NWGetDirectoryBaseExt \(page 464\)](#), [NWGetNSInfo \(page 481\)](#), [NWReadNSInfoExt \(page 528\)](#)

Name Space Structures

19

This documentation alphabetically lists the Name Space structures and describes their purpose, syntax, and fields.

- ♦ “MODIFY_DOS_INFO” on page 566
- ♦ “NW_DATA_STREAM_FAT_INFO” on page 569
- ♦ “NW_DATA_STREAM_SIZE_INFO” on page 570
- ♦ “NW_ENTRY_INFO” on page 571
- ♦ “NW_ENTRY_INFO_EXT” on page 575
- ♦ “NW_ENTRY_INFO2” on page 578
- ♦ “NW_IDX” on page 583
- ♦ “NW_MAC_TIME” on page 584
- ♦ “NW_NS_INFO” on page 585
- ♦ “NW_NS_OPEN” on page 587
- ♦ “NW_NS_OPENCREATE” on page 588
- ♦ “NW_NS_PATH” on page 591
- ♦ “SEARCH_SEQUENCE” on page 592

MODIFY_DOS_INFO

Defines the parameters for modifying an entry's DOS name space information

Service: Name Space

Defined In: nwnamspc.h

Structure

```
typedef struct
{
    nuint32    attributes;
    nuint16    createDate;
    nuint16    createTime;
    nuint32    creatorID;
    nuint16    modifyDate;
    nuint16    modifyTime;
    nuint32    modifierID;
    nuint16    archiveDate;
    nuint16    archiveTime;
    nuint32    archiverID;
    nuint16    lastAccessDate;
    nuint16    inheritanceGrantMask;
    nuint16    inheritanceRevokeMask;
    nuint32    maximumSpace;
} MODIFY_DOS_INFO;
```

Delphi Structure

```
uses calwin32

MODIFY_DOS_INFO = packed Record
    attributes : nuint32;
    createDate : nuint16;
    createTime : nuint16;
    creatorID : nuint32;
    modifyDate : nuint16;
    modifyTime : nuint16;
    modifierID : nuint32;
    archiveDate : nuint16;
    archiveTime : nuint16;
    archiverID : nuint32;
    lastAccessDate : nuint16;
    inheritanceGrantMask : nuint16;
    inheritanceRevokeMask : nuint16;
    maximumSpace : nuint32
End;
```

Fields

attributes

Specifies the attributes to the value (see [Section 20.2, “Attribute Values,”](#) on page 593).

createDate

Specifies the creation date.

createTime

Specifies the creation time.

creatorID

Specifies the creator to the specified ID.

modifyDate

Specifies the date the entry was last modified.

modifyTime

Specifies the time the entry was last modified.

modifierID

Specifies the modifier to the specified ID.

archiveDate

Specifies the date the entry was last archived.

archiveTime

Specifies the time the entry was last archived.

archiverID

Specifies the archiver of the specified ID.

lastAccessDate

Specifies the date the entry was last accessed.

inheritanceGrantMask

Specifies the inherited rights mask values (see [Section 20.4, “Inherited Rights Mask Values,”](#) on page 594).

inheritanceRevokeMask

Specifies the following TA constants:

C Value	Delphi Value	Value Name	Value Description
0x00	\$00	TA_NONE	Specifies no Reads or Writes are allowed.
0x01	\$01	TA_READ	Specifies file Reads are allowed.
0x02	\$02	TA_WRITE	Specifies file Writes are allowed.
0x08	\$08	TA_CREATE	Specifies files can be created.

C Value	Delphi Value	Value Name	Value Description
0x10	\$10	TA_DELETE	Specifies files can be deleted.
0x20	\$20	TA_OWNERSHIP	Specifies subdirectories can be created or deleted and trustee rights granted or revoked.
0x40	\$40	TA_SEARCH	Specifies the directory can be searched.
0x80	\$80	TA_MODIFY	Specifies file attributes can be modified.
0xFB	\$FB	TA_ALL	Specifies the trustee has all the above rights to the directory.

maximumSpace

Specifies the user disk restrictions (in 4 KB sizes) that may have been enabled by an administrator for the given user (optional).

NW_DATA_STREAM_FAT_INFO

Contains the FAT information for a data stream

Service: Name Space

Defined In: nwnamspc.h

Syntax

```
typedef struct
{
    nuint32    dataStreamNumber;
    nuint32    dataStreamFATBlocksSize;
} NW_DATA_STREAM_FAT_INFO;
```

Delphi Structure

```
Type
    NW_DATA_STREAM_FAT_INFO = packed Record
        dataStreamNumber : nuint32;
        dataStreamFATBlocksSize : nuint32;
    End;
```

Parameters

dataStreamNumber

Specifies the number for the data stream.

dataStreamFATBlocksSize

Specifies the size of each FAT block for the data stream.

NW_DATA_STREAM_SIZE_INFO

Contains the size information for a data stream

Service: Name Space

Defined In: nwnamspc.h

Syntax

```
typedef struct
{
    uint32_t    dataStreamNumber;
    uint32_t    dataStreamSize;
} NW_DATA_STREAM_SIZE_INFO;
```

Delphi Structure

```
Type
    NW_DATA_STREAM_SIZE_INFO = packed Record
        dataStreamNumber : uint32;
        dataStreamSize   : uint32;
    End;
```

Parameters

dataStreamNumber

Specifies the number for the data stream.

dataStreamSize

Specifies the size of the data stream.

NW_ENTRY_INFO

Holds standard name space information for an entry

Service: Name Space

Defined In: nwnamspc.h

Structure

```
typedef struct
{
    nuint32    spaceAlloc;
    nuint32    attributes;
    nuint16    flags;
    nuint32    dataStreamSize;
    nuint32    totalStreamSize;
    nuint16    numberOfStreams;
    nuint16    creationTime;
    nuint16    creationDate;
    nuint32    creatorID;
    nuint16    modifyTime;
    nuint16    modifyDate;
    nuint32    modifierID;
    nuint16    lastAccessDate;
    nuint16    archiveTime;
    nuint16    archiveDate;
    nuint32    archiverID;
    nuint16    inheritedRightsMask;
    nuint32    dirEntNum;
    nuint32    DosDirNum;
    nuint32    volNumber;
    nuint32    EADataSize;
    nuint32    EAKeyCount;
    nuint32    EAKeySize;
    nuint32    NSCreator;
    nuint8     nameLength;
    nstr8      entryName [256];
} NW_ENTRY_INFO;
```

Delphi Structure

```
uses calwin32

NW_ENTRY_INFO = packed Record
    spaceAlloc : nuint32;
    attributes : nuint32;
    flags : nuint16;
    dataStreamSize : nuint32;
    totalStreamSize : nuint32;
    numberOfStreams : nuint16;
    creationTime : nuint16;
    creationDate : nuint16;
```

```

creatorID : nuint32;
modifyTime : nuint16;
modifyDate : nuint16;
modifierID : nuint32;
lastAccessDate : nuint16;
archiveTime : nuint16;
archiveDate : nuint16;
archiverID : nuint32;
inheritedRightsMask : nuint16;
dirEntNum : nuint32;
DosDirNum : nuint32;
volNumber : nuint32;
EADataSize : nuint32;
EAKeyCount : nuint32;
EAKeySize : nuint32;
NSCreator : nuint32;
nameLength : nuint8;
entryName : Array[0..255] Of nstr8
End;

```

Fields

spaceAlloc

Specifies the space allocated to the data stream. `IM_SPACE_ALLOC` in `returnEntryInfo` mask.

attributes

Specifies the entry's attributes (see [Section 20.2, "Attribute Values,"](#) on page 593).

flags

Specifies data used internally.

dataStreamSize

Specifies the size of the data stream. `IM_SIZE` in `returnEntryInfo` mask.

totalStreamSize

Specifies the total size of streams associated with the entry. `IM_TOTAL_SIZE` in `returnEntryInfo` mask.

numberOfStreams

Specifies the number of streams associated with the entry.

creationTime

Specifies when the entry was created. `IM_CREATION` in `returnEntryInfo` mask (see [Section 20.9, "Time Values,"](#) on page 597).

creationDate

Specifies the date the entry was created (see [Section 20.3, "Date Values,"](#) on page 594).

creatorID

Specifies the object creating the entry.

modifyTime

Specifies the time the entry was last modified. IM_MODIFY in `returnEntryInfo` mask (see [Section 20.9, “Time Values,” on page 597](#)).

modifyDate

Specifies the date the entry was last modified (see [Section 20.3, “Date Values,” on page 594](#)).

modifierID

Specifies the ID of the object that last modified the entry.

lastAccessDate

Specifies the date the entry was last accessed (see [Section 20.3, “Date Values,” on page 594](#)).

archiveTime

Specifies the time the entry was last archived (see [Section 20.9, “Time Values,” on page 597](#)).

archiveDate

Specifies the date the entry was last archived (see [Section 20.3, “Date Values,” on page 594](#)).

archiverID

Specifies the ID of the object last archiving the entry.

inheritedRightsMask

Specifies the entry’s inherited rights mask. IM_RIGHTS in `returnEntryInfo` mask. A mask of the following:

C Value	Delphi Value	Value Name	Value Description
0x00	\$00	TA_NONE	Specifies no Reads or Writes are allowed.
0x01	\$01	TA_READ	Specifies file Reads are allowed.
0x02	\$02	TA_WRITE	Specifies file Writes are allowed.
0x08	\$08	TA_CREATE	Specifies files can be created.
0x10	\$10	TA_DELETE	Specifies files can be deleted.
0x20	\$20	TA_OWNERSHIP	Specifies subdirectories can be created or deleted and trustee rights granted or revoked.
0x40	\$40	TA_SEARCH	Specifies the directory can be searched.
0x80	\$80	TA_MODIFY	Specifies file attributes can be modified.
0xFB	\$FB	TA_ALL	Specifies the trustee has all the above rights to the directory.

dirEntNum

Specifies the directory entry number. IM_DIRECTORY in `returnEntryInfo` mask.

DosDirNum

Specifies the DOS directory entry number.

volNumber

Specifies the number of the volume that contains the entry.

EADataSize

Specifies the data size of the entry's extended attribute. `IM_EA` in `returnEntryInfo` mask.

EAKeyCount

Specifies the key count for the entry's extended attribute.

EAKeySize

Specifies the size of the entry's extended attribute key.

NSCreator

Specifies the name space the entry was originally created in. `IM_OWNING_NAMESPACE` in `returnEntryInfo` mask (see [Section 20.5, "Name Space Flag Values,"](#) on page 595).

nameLength

Specifies the length of the entry's name. `IM_NAME` in `returnEntryInfo` mask.

entryName

Specifies the entry's name.

NW_ENTRY_INFO_EXT

Holds standard name space information for an entry and uses UTF-8 strings.

Service: Name Space

Defined In: nwnamspc.h

Structure

```
typedef struct
{
    nuint32    spaceAlloc;
    nuint32    attributes;
    nuint16    flags;
    nuint32    dataStreamSize;
    nuint32    totalStreamSize;
    nuint16    numberOfStreams;
    nuint16    creationTime;
    nuint16    creationDate;
    nuint32    creatorID;
    nuint16    modifyTime;
    nuint16    modifyDate;
    nuint32    modifierID;
    nuint16    lastAccessDate;
    nuint16    archiveTime;
    nuint16    archiveDate;
    nuint32    archiverID;
    nuint16    inheritedRightsMask;
    nuint32    dirEntNum;
    nuint32    DosDirNum;
    nuint32    volNumber;
    nuint32    EADataSize;
    nuint32    EAKeyCount;
    nuint32    EAKeySize;
    nuint32    NSCreator;
    nuint8     nameLength;
    nstr8      entryName [766];
} NW_ENTRY_INFO_EXT;
```

Fields

spaceAlloc

Specifies the space allocated to the data stream. `IM_SPACE_ALLOC` in `returnEntryInfo` mask.

attributes

Specifies the entry's attributes (see [Section 20.2, "Attribute Values,"](#) on page 593).

flags

Specifies data used internally.

dataStreamSize

Specifies the size of the data stream. IM_SIZE in returnEntryInfo mask.

totalStreamSize

Specifies the total size of streams associated with the entry. IM_TOTAL_SIZE in returnEntryInfo mask.

numberOfStreams

Specifies the number of streams associated with the entry.

creationTime

Specifies when the entry was created. IM_CREATION in returnEntryInfo mask (see [Section 20.9, “Time Values,” on page 597](#)).

creationDate

Specifies the date the entry was created (see [Section 20.3, “Date Values,” on page 594](#)).

creatorID

Specifies the object creating the entry.

modifyTime

Specifies the time the entry was last modified. IM_MODIFY in returnEntryInfo mask (see [Section 20.9, “Time Values,” on page 597](#)).

modifyDate

Specifies the date the entry was last modified (see [Section 20.3, “Date Values,” on page 594](#)).

modifierID

Specifies the ID of the object that last modified the entry.

lastAccessDate

Specifies the date the entry was last accessed (see [Section 20.3, “Date Values,” on page 594](#)).

archiveTime

Specifies the time the entry was last archived (see [Section 20.9, “Time Values,” on page 597](#)).

archiveDate

Specifies the date the entry was last archived (see [Section 20.3, “Date Values,” on page 594](#)).

archiverID

Specifies the ID of the object last archiving the entry.

inheritedRightsMask

Specifies the entry’s inherited rights mask. IM_RIGHTS in returnEntryInfo mask. A mask of the following:

C Value	Value Name	Value Description
0x00	TA_NONE	Specifies no Reads or Writes are allowed.
0x01	TA_READ	Specifies file Reads are allowed.

C Value	Value Name	Value Description
0x02	TA_WRITE	Specifies file Writes are allowed.
0x08	TA_CREATE	Specifies files can be created.
0x10	TA_DELETE	Specifies files can be deleted.
0x20	TA_OWNERSHIP	Specifies subdirectories can be created or deleted and trustee rights granted or revoked.
0x40	TA_SEARCH	Specifies the directory can be searched.
0x80	TA_MODIFY	Specifies file attributes can be modified.
0xFB	TA_ALL	Specifies the trustee has all the above rights to the directory.

dirEntNum

Specifies the directory entry number. IM_DIRECTORY in returnEntryInfo mask.

DosDirNum

Specifies the DOS directory entry number.

volNumber

Specifies the number of the volume that contains the entry.

EADataSize

Specifies the data size of the entry's extended attribute. IM_EA in returnEntryInfo mask.

EAKeyCount

Specifies the key count for the entry's extended attribute.

EAKeySize

Specifies the size of the entry's extended attribute key.

NSCreator

Specifies the name space the entry was originally created in. IM_OWNING_NAMESPACE in returnEntryInfo mask (see [Section 20.5, "Name Space Flag Values,"](#) on page 595).

nameLength

Specifies the length of the entry's name. IM_NAME in returnEntryInfo mask.

entryName

Specifies the entry's name, using UTF-8 characters.

NW_ENTRY_INFO2

Holds standard name space information for an entry

Service: Name Space

Defined In: nwnamspc.h

Structure

```
typedef struct
{
    nuint32          spaceAlloc;
    nuint32          attributes;
    nuint16          flags;
    nuint32          dataStreamSize;
    nuint32          totalStreamSize;
    nuint16          numberOfStreams;
    nuint32          EADataSize;
    nuint32          EAKeyCount;
    nuint32          EAKeySize;
    nuint16          archiveTime;
    nuint16          archiveDate;
    nuint32          archiverID;
    nuint16          modifyTime;
    nuint16          modifyDate;
    nuint32          modifierID;
    nuint16          lastAccessDate;
    nuint16          creationTime;
    nuint16          creationDate;
    nuint32          creatorID;
    nuint32          NSCreator;
    nuint32          dirEntNum;
    nuint32          DosDirNum;
    nuint32          volNumber;
    nuint16          inheritedRightsMask;
    nuint16          currentReferenceID;
    nuint32          NSFileAttributes;
    nuint32          numberOfDataStreamFATInfo;
    NW_DATA_STREAM_FAT_INFO dataStreamFATInfo[3];
    nuint32          numberOfDataStreamSizeInfo;
    NW_DATA_STREAM_SIZE_INFO dataStreamSizeInfo[3];
    nint32           secondsRelativeToTheYear2000;
    nuint8           DOSNameLen;
    nstr8            DOSName[13];
    nuint32          flushTime;
    nuint32          parentBaseID;
    nuint8           MacFinderInfo[32];
    nuint32          siblingCount;
    nuint32          effectiveRights;
    NW_MAC_TIME      MacTime;
    nuint16          lastAccessedTime;
    nuint8           nameLength;
}
```

```

    nstr8                entryName[256];
} NW_ENTRY_INFO2;

```

Delphi Structure

```

NW_ENTRY_INFO2 = packed Record
    spaceAlloc :nuint32;
    attributes : nuint32;
    flags      : nuint16;
    dataStreamSize :nuint32;
    totalStreamSize:nuint32;
    numberOfStreams : nuint16;
    EADataSize :nuint32;
    EAKeyCount:nuint32;
    EAKeySize :nuint32;
    archiveTime : nuint16;
    archiveDate : nuint16;
    archiverID : nuint32;
    modifyTime : nuint16;
    modifyDate :nuint16;
    modifierID :nuint32;
    lastAccessDate :nuint16;
    creationTime :nuint16;
    creationDate :nuint16;
    creatorID : nuint32;
    NSCreator : nuint32;
    dirEntNum :nuint32;
    DosDirNum : nuint32;
    volNumber :nuint32;
    inheritedRightsMask :nuint16;
    currentReferenceID:nuint16;
    NSFileAttributes : nuint32;
    numberOfDataStreamFATInfo :nuint32;
    dataStreamFATInfo:Array[1..3]of NW_DATA_STREAM_FAT_INFO;
    numberOfDataStreamSizeInfo :nuint32;
    dataStreamSizeInfo :Array[1..3]of
NW_DATA_STREAM_SIZE_INFO;
    secondsRelativeToTheYear2000 : nint32;
    DOSNameLen : nuint8;
    DOSName :Array[1..13] of nstr8;
    flushTime : nuint32;
    parentBaseID : nuint32;
    MacFinderInfo :Array[1..32] of nuint8;
    siblingCount : nuint32;
    effectiveRights : nuint32;
    MacTime : NW_MAC_TIME;
    lastAccessedTime :nuint16;
    nameLength : nuint8;
    entryName : Array[0..255] of nstr8;
end;

```

Fields

spaceAlloc

Specifies the space allocated to the data stream (see [Section 20.6, “Basic Return Mask Values,”](#) on page 595).

attributes

Specifies the entry’s attributes (see [Section 20.2, “Attribute Values,”](#) on page 593).

flags

Specifies data used internally.

dataStreamSize

Specifies the size of the data stream.

totalStreamSize

Specifies the total size of streams associated with the entry.

numberOfStreams

Specifies the number of streams associated with the entry.

EADataSize

Specifies the data size of the entry’s extended attribute.

EAKeyCount

Specifies the key count for the entry’s extended attribute.

EAKeySize

Specifies the size of the entry’s extended attribute key.

archiveTime

Specifies the time the entry was last archived (see [Section 20.9, “Time Values,”](#) on page 597).

archiveDate

Specifies the date the entry was last archived (see [Section 20.3, “Date Values,”](#) on page 594).

archiverID

Specifies the ID of the object last archiving the entry.

modifyTime

Specifies the time the entry was last modified (see [Section 20.9, “Time Values,”](#) on page 597).

modifyDate

Specifies the date the entry was last modified (see [Section 20.3, “Date Values,”](#) on page 594).

modifierID

Specifies the ID of the object that last modified the entry.

lastAccessDate

Specifies the date the entry was last accessed (see [Section 20.3, “Date Values,”](#) on page 594).

creationTime

Specifies when the entry was created (see [Section 20.9, “Time Values,”](#) on page 597).

creationDate

Specifies the date the entry was created (see [Section 20.3, “Date Values,”](#) on page 594).

creatorID

Specifies the object creating the entry.

NSCreator

Specifies the name space the entry was originally created in (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

dirEntNum

Specifies the directory entry number.

DosDirNum

Specifies the DOS directory entry number.

volNumber

Specifies the number of the volume that contains the entry.

inheritedRightsMask

Specifies the entry’s inherited rights mask (see [Section 20.4, “Inherited Rights Mask Values,”](#) on page 594).

currentReferenceID

Specifies the change count information.

NSFileAttributes

Specifies the name space file attributes.

numberOfDataStreamFATInfo

Specifies the number of valid NW_DATA_STREAM_FAT_INFO structures.

dataStreamFATInfo

Points to NW_DATA_STREAM_FAT_INFO.

numberOfDataStreamSizeInfo

Specifies the number of valid NW_DATA_STREAM_SIZE_INFO structures.

dataStreamSizeInfo

Points to NW_DATA_STREAM_SIZE_INFO.

secondsRelativeToTheYear2000

Specifies the number of seconds until (negative values) or after (positive values) 12:00 a.m. on January 1, 2000.

DOSNameLen

Specifies the length of the DOS name.

DOSName

Specifies the DOS name.

flushTime

Specifies the flush time for the scanned item.

parentBaseID

Specifies the parent directory base number for a file or subdirectory.

MacFinderInfo

Specifies the MAC finder information for a scanned item.

siblingCount

Specifies the number of siblings in a subdirectory.

effectiveRights

Specifies the effective rights for a file.

MacTime

Points to NW_MAC_TIME.

lastAccessedTime

Specifies the time the file was last accessed.

nameLength

Specifies the length of the entry's name.

entryName

Specifies the entry's name.

NW_IDX

Receives the directory base for an entry in a specified name space

Service: Name Space

Defined In: nwnamspc.h

Structure

```
typedef struct
{
    nuint8    volNumber ;
    nuint8    srcNameSpace ;
    nuint32   srcDirBase ;
    nuint8    dstNameSpace ;
    nuint32   dstDirBase ;
} NW_IDX;
```

Delphi Structure

```
uses calwin32

NW_IDX = packed Record
    volNumber : nuint8;
    srcNameSpace : nuint8;
    srcDirBase : nuint32;
    dstNameSpace : nuint8;
    dstDirBase : nuint32
End;
```

Fields

volNumber

Specifies the volume number.

srcNameSpace

Specifies the name space of source (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

srcDirBase

Specifies the directory base of source.

dstNameSpace

Specifies the name space changing to (see [Section 20.5, “Name Space Flag Values,” on page 595](#)).

dstDirBase

Specifies the directory base of the entry in the new name space.

NW_MAC_TIME

Contains information about the MAC time for the scanned item

Service: Name Space

Defined In: nwnamspc.h

Syntax

```
typedef struct
{
    uint32_t    MACCreateTime;
    uint32_t    MACBackupTime;
} NW_MAC_TIME;
```

Delphi Structure

```
Type
    NW_MAC_TIME = packed Record
        MACCreateTime : uint32;
        MACBackupTime : uint32;
    End;
```

Parameters

MACCreateTime

Specifies the creation time for a MAC file.

MACBackupTime

Specifies the backup time for a MAC file.

NW_NS_INFO

Handles the information bit masks used to read name space-specific information

Service: Name Space

Defined In: nwnamspc.h

Structure

```
typedef struct
{
    nuint32    NSInfoBitMask ;
    nuint32    fixedBitMask ;
    nuint32    reservedBitMask ;
    nuint32    extendedBitMask ;
    nuint16    fixedBitsDefined ;
    nuint16    reservedBitDefined ;
    nuint16    extendedBitsDefined ;
    nuint32    fieldsLenTable [32];
    nuint8     hugeStateInfo [16];
    nuint32    hugeDataLength ;
} NW_NS_INFO;
```

Delphi Structure

```
uses calwin32

NW_NS_INFO = packed Record
    NSInfoBitMask : nuint32;
    fixedBitMask : nuint32;
    reservedBitMask : nuint32;
    extendedBitMask : nuint32;
    fixedBitsDefined : nuint16;
    reservedBitsDefined : nuint16;
    extendedBitsDefined : nuint16;
    fieldsLenTable : Array[0..31] Of nuint32;
    hugeStateInfo : Array[0..15] Of nuint8;
    hugeDataLength : nuint32
End;
```

Fields

NSInfoBitMask

Specifies a bit mask with the following definitions:

C Value	Delphi Value	Value Name
0x0002L	\$0002	DM_ATTRIBUTES
0x0004L	\$0004	DM_CREATE_DATE

C Value	Delphi Value	Value Name
0x0008L	\$0008	DM_CREATE_TIME
0x0010L	\$0010	DM_CREATOR_ID
0x0020L	\$0020	DM_ARCHIVE_DATE
0x0040L	\$0040	DM_ARCHIVE_TIME
0x0080L	\$0080	DM_ARCHIVER_ID
0x0100L	\$0100	DM_MODIFY_DATE
0x0200L	\$0200	DM_MODIFY_TIME
0x0400L	\$0400	DM_MODIFIER_ID
0x0800L	\$0800	DM_LAST_ACCESS_DATE
0x1000L	\$1000	DM_INHERITED_RIGHTS_MASK
0x2000L	\$2000	DM_MAXIMUM_SPACE

fixedBitMask

Specifies a bit mask representing fixed (sized) information.

reservedBitMask

Specifies a bit mask representing information stored as a length-preceded array. The first byte indicates the length.

extendedBitMask

Specifies a bit mask representing information stored as a length-preceded string with the first 2 bytes indicating the length.

fixedBitsDefined

Specifies a value indicating how many bits are defined within `fixedBitMask`.

reservedBitDefined

Specifies a value indicating how many bits are defined within `reservedBitMask`.

extendedBitsDefined

Specifies a value indicating how many bits are defined within `extendedBitMask`.

fieldsLenTable

Specifies the length of the information relative to any of the three bit masks. receives values that indicate how many bits are defined within `reservedBitMask`.

hugeStateInfo

Is used only by NFS.

hugeDataLength

Specifies the length of the data that is returned in the reply buffer.

NW_NS_OPEN

Is defined to be the same as the [NW_NS_OPENCREATE \(page 588\)](#) structure

Service: Name Space

Defined In: nwnamspc.h

NW_NS_OPENCREATE

Defines the parameters for opening/creating a data stream in a specified name space

Service: Name Space

Defined In: nwnamspc.h

Structure

```
typedef struct
{
    nuint8    openCreateMode ;
    nuint16   searchAttributes ;
    nuint32   reserved ;
    nuint32   createAttributes ;
    nuint16   accessRights ;
    nuint32   NetWareHandle ;
    nuint8    openCreateAction ;
} NW_NS_OPENCREATE
```

Delphi Structure

```
uses calwin32

NW_NS_OPENCREATE = packed Record
    openCreateMode : nuint8;
    searchAttributes : nuint16;
    reserved : nuint32;
    createAttributes : nuint32;
    accessRights : nuint16;
    NetWareHandle : nuint32;
    openCreateAction : nuint8
End;
```

Fields

openCreateMode

Specifies whether to create, replace, or open an entry (directories can only be created). Open/Create modes use the OC_MODE_ constants listed below:

C Value	Delphi Value	Value Name
0x01	\$01	OC_MODE_OPEN
0x02	\$02	OC_MODE_TRUNCATE
0x02	\$02	OC_MODE_REPLACE
0x08	\$08	OC_MODE_CREATE

searchAttributes

Specifies the attributes to use in the search (see [Section 20.8, “Search Attributes Values,”](#) on [page 597](#)).

reserved

Is reserved for future use.

createAttributes

Specifies the attributes to set in the DOS name space (see [Section 20.2, “Attribute Values,”](#) on [page 593](#)).

accessRights

Specifies the desired access rights (see [Section 20.1, “Access Right Values,”](#) on [page 593](#)).

NWHandle

Specifies a four-byte NetWare handle.

openCreateAction

Specifies the result of a successful open/create. Uses the OC_ACTION_ constants listed below:

C Value	Delphi Value	Value Name
0x01	\$01	OC_ACTION_NONE
0x01	\$01	OC_ACTION_OPEN
0x02	\$02	OC_ACTION_CREATE
0x04	\$04	OC_ACTION_TRUNCATE
0x04	\$04	OC_ACTION_REPLACE

Remarks

To create a file, the `accessRights` field is used as an access rights mask and must be set to `AR_READ` and/or `AR_WRITE`. If neither are used, the `NW_NS_OPENCREATE` structure sets both. Use the AR constants listed below:

To create a directory, the `accessRights` field is used as an inherited rights mask and has the following bits:

0	Read Existing File Bit
1	Write Existing File Bit
2	Old Open Existing File Bit
3	Create New Entry Bit
4	Delete Existing Bit
5	Change Access Control Bit
6	See Files Bit

7	Modify Entry Bit
8	Supervisor Privileges Bit
9-15	not set

NW_NS_PATH

Defines parameters for returning an entry's path with in a specified name space

Service: Name Space

Defined In: nwnamspc.h

Structure

```
typedef struct
{
    pustr8    srcPath ;
    pustr8    dstPath ;
    nuint16   dstPathSize ;
} NW_NS_PATH;
```

Delphi Structure

```
uses calwin32

NW_NS_PATH = packed Record
    srcPath : pustr8;
    dstPath : pustr8;
    dstPathSize : nuint16
End;
```

Fields

srcPath

Points to a valid path. When this structure used with the `NWGetNSPathExt` function, the characters in the path string must be UTF-8.

dstPath

Points to a buffer to receive the full name space path. When this structure used with the `NWGetNSPathExt` function, the destination path is returned in UTF-8 characters.

dstPathSize

Specifies the length of new path buffer. The new path buffer should be long enough to hold the longest path possible for `destNameSpace` plus 2 extra bytes for working space.

Remarks

The [NWGetNSPath \(page 489\)](#) and [NWGetNSPathExt \(page 491\)](#) functions use this structure. The `NWGetNSPath` function gets and returns strings in the local code page; the `NWGetNSPathExt` gets and returns strings in UTF-8 on NSS volumes.

SEARCH_SEQUENCE

Defines information for managing a search operation across multiple requests

Service: Name Space

Defined In: nwnamspc.h

Structure

```
typedef struct
{
    nuint8    volNumber ;
    nuint32   dirNumber ;
    nuint32   searchDirNumber ;
} SEARCH_SEQUENCE;
```

Delphi Structure

```
uses calwin32

SEARCH_SEQUENCE = packed Record
    volNumber : nuint8;
    dirNumber : nuint32;
    searchDirNumber : nuint32
End;
```

Fields

volNumber

Specifies the volume number.

dirNumber

Specifies the directory entry number for the directory.

searchDirNumber

Specifies the directory number to search. Set to a 0xFFFFFFFF on the first call. After that, `searchDirNumber` is managed internally.

Name Space Values

20

This documentation describes the values associated with Name Space.

20.1 Access Right Values

The following are access right values:

C Value	Delphi Value	Value Name
0x0001	\$0001	AR_READ
0x0002	\$0002	AR_WRITE
0x0001	\$0001	AR_READ_ONLY
0x0002	\$0002	AR_WRITE_ONLY
0x0004	\$0004	AR_DENY_READ
0x0008	\$0008	AR_DENY_WRITE
0x0010	\$0010	AR_COMPATIBILITY
0x0040	\$0040	AR_WRITE_THROUGH
0x0100	\$0100	AR_OPEN_COMPRESSED

AR_OPEN_COMPRESSED cannot be used with NWFPOpenFileFork since this function only accepts an 8-bit constant for the `accessMode` parameter.

20.2 Attribute Values

The following are attribute values:

C Value	Delphi Value	Value Name
0x00000000L	\$00000000	A_NORMAL
0x00000001L	\$00000001	A_READ_ONLY
0x00000002L	\$00000002	A_HIDDEN
0x00000004L	\$00000004	A_SYSTEM
0x00000008L	\$00000008	A_EXECUTE_ONLY
0x00000010L	\$00000010	A_DIRECTORY
0x00000020L	\$00000020	A_NEEDS_ARCHIVED
0x00000080L	\$00000080	A_SHAREABLE
0x00001000L	\$00001000	A_TRANSACTIONAL
0x00002000L	\$00002000	A_INDEXED

C Value	Delphi Value	Value Name
0x0004000L	\$0004000	A_READ_AUDIT
0x0008000L	\$0008000	A_WRITE_AUDIT
0x0010000L	\$0010000	A_IMMEDIATE_PURGE
0x0020000L	\$0020000	A_RENAME_INHIBIT
0x0040000L	\$0040000	A_DELETE_INHIBIT
0x0080000L	\$0080000	A_COPY_INHIBIT
0x0400000L	\$0400000	A_FILE_MIGRATED
0x0800000L	\$0800000	A_DONT_MIGRATE
0x02000000L	\$02000000	A_IMMEDIATE_COMPRESS
0x04000000L	\$04000000	A_FILE_COMPRESSED
0x08000000L	\$08000000	A_DONT_COMPRESS
0x20000000L	\$20000000	A_CANT_COMPRESS

20.3 Date Values

From the least significant byte to the most significant byte:

The first 5 bits indicate the day, from 1-31.

The next 4 bits indicate the month, from 1-12.

The last 7 bits indicate the year, with 0 = 1980 and 20 = 2000.

20.4 Inherited Rights Mask Values

`inheritanceGrantMask` and `inheritedRightsMask` can have the following values:

C Value	Delphi Value	Value Name	Value Description
0x00	\$00	TA_NONE	Specifies no Reads or Writes are allowed.
0x01	\$01	TA_READ	Specifies file Reads are allowed.
0x02	\$02	TA_WRITE	Specifies file Writes are allowed.
0x08	\$08	TA_CREATE	Specifies files can be created.
0x10	\$10	TA_DELETE	Specifies files can be deleted.
0x20	\$20	TA_OWNERSHIP	Specifies subdirectories can be created or deleted and trustee rights granted or revoked.
0x40	\$40	TA_SEARCH	Specifies the directory can be searched.
0x80	\$80	TA_MODIFY	Specifies file attributes can be modified.
0xFB	\$FB	TA_ALL	Specifies the trustee has all the above rights to the directory.

20.5 Name Space Flag Values

The following table lists the values used in setting and retrieving name space information.

Value	Constant	Description
0	NW_NS_DOS	DOS name space.
1	NW_NS_MAC	Macintosh name space.
2	NW_NS_NFS	NFS name space.
3	NW_NS_FTAM	FTAM name space.
4	NW_NS_LONG	Windows 32-bit name space. This flag is the same as NW_NS_OS2 and can be used for the OS/2 name space.

20.6 Basic Return Mask Values

See [Section 20.7, “Extended Return Mask Values,” on page 596](#) for the extended values.

Return mask parameters can have the following values:

C Value	Delphi Value	Value Name
0x0001L	\$0001	IM_NAME (3.x and above)—corresponds to <code>nameLength</code> and <code>entryName</code> and is always returned by <code>NW_ENTRY_INFO2</code> .
0x0001L	\$0001	IM_ENTRY_NAME
0x0002L	\$0002	IM_SPACE_ALLOCATED (3.x and above)—corresponds to <code>spaceAlloc</code> in <code>NW_ENTRY_INFO2</code> .
0x0004L	\$0004	IM_ATTRIBUTES (3.x and above)—corresponds to <code>attributes</code> and <code>flags</code> in <code>NW_ENTRY_INFO2</code> .
0x0008L	\$0008	IM_SIZE (3.x and above)—corresponds to <code>dataStreamSize</code> in <code>NW_ENTRY_INFO2</code> .
0x0010L	\$0010	IM_TOTAL_SIZE (3.x and above)—corresponds to <code>totalStreamSize</code> and <code>numberOfStreams</code> in <code>NW_ENTRY_INFO2</code> .
0x0020L	\$0020	IM_EA (3.x and above)—corresponds to <code>EADataSize</code> , <code>EAKeyCount</code> , and <code>EAKeySize</code> in <code>NW_ENTRY_INFO2</code> .
0x0040L	\$0040	IM_ARCHIVE (3.x and above)—corresponds to <code>archiveTime</code> , <code>archiveDate</code> , and <code>archiverID</code> in <code>NW_ENTRY_INFO2</code> .
0x0080L	\$0080	IM_MODIFY (3.x and above)—corresponds to <code>modifyTime</code> , <code>modifyDate</code> , <code>modifierID</code> , and <code>lastAccessDate</code> in <code>NW_ENTRY_INFO2</code> .
0x0100L	\$0100	IM_CREATION (3.x and above)—corresponds to <code>creationTime</code> , <code>creationDate</code> , and <code>creatorID</code> in <code>NW_ENTRY_INFO2</code> .
0x0200L	\$0200	IM_OWNING_NAMESPACE (3.x and above)—corresponds to <code>NSCreator</code> in <code>NW_ENTRY_INFO2</code> .
0x0400L	\$0400	IM_DIRECTORY (3.x and above)—corresponds to <code>dirEntNum</code> , <code>DosDirNum</code> , and <code>volNumber</code> in <code>NW_ENTRY_INFO2</code> .

C Value	Delphi Value	Value Name
0x0800L	\$0800	IM_RIGHTS (3.x and above)—corresponds to <code>inheritedRightsMask</code> in <code>NW_ENTRY_INFO2</code> .
0x0FEDL	\$0FED	IM_ALMOST_ALL
0x0FFFL	\$0FFF	IM_ALL

20.7 Extended Return Mask Values

See [Section 20.6, “Basic Return Mask Values,”](#) on page 595 for the basic values.

Parameters in functions with extended return mask functionality can have the values that follow in addition to basic return mask values. Successful use of these values is limited to functions on NetWare 4.10 or higher.

C Value	Delphi Value	Value Name
0x1000L	\$1000	IM_REFERENCE_ID (4.1x and above)—corresponds to <code>currentReferenceID</code> in <code>NW_ENTRY_INFO2</code> .
0x2000L	\$2000	IM_NS_ATTRIBUTES (4.1x and above)—corresponds to <code>NSFileAttributes</code> in <code>NW_ENTRY_INFO2</code> .
0x4000L	\$4000	IM_DATASTREAM_SIZES or IM_DATASTREAM_ACTUAL (4.1x and above)—corresponds to <code>numberOfDataStreamFATInfo</code> and <code>dataStreamFATInfo[3]</code> in <code>NW_ENTRY_INFO2</code> . <code>numberOfDataStreamFATInfo</code> specifies how many items were actually returned in <code>dataStreamFATInfo[3]</code> .
0x8000L	\$8000	IM_DATASTREAM_LOGICAL
0x00010000L	\$00010000	IM_LASTUPDATEDINSECONDS (4.1x and above)—corresponds to <code>secondsRelativeToTheYear2000</code> in <code>NW_ENTRY_INFO2</code> .
0x00020000L	\$00020000	IM_DOSNAME (4.1x and above)—corresponds to <code>DOSNameLen</code> and <code>DOSName[13]</code> in <code>NW_ENTRY_INFO2</code> .
0x00040000L	\$00040000	IM_FLUSHTIME (4.1x and above)—corresponds to <code>flushTime</code> in <code>NW_ENTRY_INFO2</code> .
0x00080000L	\$00080000	IM_PARENTBASEID (4.1x and above)—corresponds to <code>parentBaseID</code> in <code>NW_ENTRY_INFO2</code> .
0x00100000L	\$00100000	IM_MACFINDER (4.1x and above)—corresponds to <code>MacFinderInfo[32]</code> in <code>NW_ENTRY_INFO2</code> .
0x00200000L	\$00200000	IM_SIBLINGCOUNT (4.1x and above)—corresponds to <code>siblingCount[32]</code> in <code>NW_ENTRY_INFO2</code> and applies only to a directory entry. For files, zero is returned. This is the number of entries in the directory (excluding "." and "..").
0x00400000L	\$00400000	IM_EFFECTIVERIGHTS (4.1x and above)—corresponds to <code>effectiveRights</code> in <code>NW_ENTRY_INFO2</code> .
0x00800000L	\$00800000	IM_MACTIME (4.1x and above)—corresponds to <code>MacTime</code> in <code>NW_ENTRY_INFO2</code> .

C Value	Delphi Value	Value Name
0x01000000 L	\$01000000	IM_LASTACCESSTIME (5.x and above)—corresponds to <code>lastAccessedTime</code> in <code>NW_ENTRY_INFO2</code> .
0x01FFF000 L	\$01FFF000	IM_EXTENDED_ALL is used to return all the extended information corresponding to bits 12-24 in <code>retInfoMask</code> .
0x40000000 L	\$40000000	IM_NSS_LARGE_SIZES
0x80000000 L	\$80000000	IM_COMPRESSED_INFO
0x80000000 L	\$80000000	IM_NS_SPECIFIC_INFO (4.1x and above)—corresponds to <code>numberOfDataStreamSizeInfo</code> and <code>dataStreamSizeInfo[3]</code> in <code>NW_ENTRY_INFO2</code> . <code>numberOfDataStreamSizeInfo</code> specifies how many items were actually returned in <code>dataStreamSizeInfo[3]</code> .

20.8 Search Attributes Values

The following are search attribute values:

C Value	Delphi Value	Value Name
0x0000	\$0000	SA_NORMAL
0x0002	\$0002	SA_HIDDEN
0x0004	\$0004	SA_SYSTEM
0x0010	\$0010	SA_SUBDIR_ONLY
0x8000	\$8000	SA_SUBDIR_FILES
0x8006	\$8006	SA_ALL

20.9 Time Values

From the least significant byte to the most significant byte:

The first 5 bits indicate the number of 2-second intervals, from 0-29 so that 59 and 60 seconds are both indicated by 29.

The next 6 bits indicate the minute, from 0-59.

The last 5 bits indicate the hour, from 0-23.

Path and Drive Concepts

21

This documentation describes Path and Drive, its functions, and features.

Path and Drive controls the workstation's relationship to the network. Specifically, it configures the workstation environment by managing network drive mappings. However, it does not formulate requests for NetWare servers.

21.1 Path Parameters

NWGetDriveStatus and NWGetDriveStatusConnRef return path information in four path parameters.

pathFormat expects one of the following four constants:

```
NW_FORMAT_NETWARE      0
NW_FORMAT_SERVER_VOLUME 1
NW_FORMAT_DRIVE        2
NW_FORMAT_UNC          3
```

For the NetWare, Server Volume, and UNC constants, the value of the fullPath parameter will equal the value of the rootPath parameter, plus a backslash character, plus the value of the relPath parameter. For the Drive constant, the value of the fullPath parameter will equal the value of the rootPath parameter plus the value of the relPath parameter (without adding a backslash character).

The following tables explain what will be returned in each of the path output parameters for each of the pathFormat constants.

Assume you are in dir2 and drive letter Q is root mapped to the following:

```
server\volume:dir1
```

	rootPath	relPath	fullPath
NetWare	volume:dir1	dir2	volume:dir1\dir2
Server Volume	server\volume:dir1	dir2	server\volume:dir1\dir2
Drive	Q:\	dir2	Q:\dir1\dir2
UNC	\\server\volume\dir1	dir2	\\server\volume\dir1\dir2

Assume you are in dir1\dir2 and drive letter Q is root mapped to the following:

```
server\volume:
```

	rootPath	relPath	fullPath
NetWare	volume:	dir1\dir2	volume:\dir1\dir2
Server Volume	server\volume:	dir1\dir2	server\volume:\dir1\dir2
Drive	Q:\	dir1\dir2	Q:\dir1\dir2

	rootPath	relPath	fullPath
UNC	\\server\volume	dir1\dir2	\\server\volume\dir1\dir2

The `status` parameter returns a bit mask indicating if a drive is a local and/or network drive:

C Value	Delphi Value	Value Name
0x0000	\$0000	NW_UNMAPPED_DRIVE
0x0000	\$0000	NW_FREE_DRIVE
0x0400	\$0400	NW_CDROM_DRIVE
0x0800	\$0800	NW_LOCAL_FREE_DRIVE
0x1000	\$1000	NW_LOCAL_DRIVE
0x2000	\$2000	NW_NETWORK_DRIVE
0x4000	\$4000	NW_PNW_DRIVE
0x8000	\$8000	NW_NETWARE_DRIVE

21.2 Network Drive Functions

Path and Drive services include functions that manage network drive mappings. The following are the functions most commonly used:

- ◆ [NWGetDriveStatus \(page 614\)](#) returns information about a drive mapping.
- ◆ [NWSetDriveBase \(page 627\)](#) sets a drive mapping.
- ◆ [NWDeleteDriveBase \(page 608\)](#) deletes a drive mapping.

These functions map network drives, return drive information, perform parsing on path strings, and access the Netx search drive vector. It is possible that a specific client supports only a subset of these functions.

NWDeleteDriveBase	Deletes a network drive mapping.
NWGetDriveInformation	Returns information about the specified drive.
NWGetDriveStatus	Returns the status of the specified drive and, optionally, the associated connection and its path in various formats.
NWGetFirstDrive	Returns the first non-local drive.
NWParseNetWarePath	Parses a path and returns the connection handle, directory handle, and new path to be used by subsequent NetWare requests.
NWParsePath	Parses a path string.
NWSetDriveBase	Maps the target drive to the specified directory path.
NWStripServerOffPath	Parses a server or volume path, copies the server name to the buffer specified by server, and returns a pointer to the volume path.

This documentation describes common tasks associated with Path and Drive.

22.1 Listing Network Drives

The following steps allow you to determine if the specified drive is a NetWare® drive:

- 1 Initialize the client libraries by calling `NWCallsInit` (Client Management).
- 2 For each of the drives, 1 through 26, call `NWGetDriveStatus` (page 614) and check the `status` parameter to determine if the drive is a NetWare drive.

22.2 Mapping Network Drives

The following steps allow you to associate a NetWare® path with a client's drive. For an example, see "Mapping a Network Drive Example" on page 601.

- 1 Determine the path to be mapped to and the drive letter that is to be associated with the path.
- 2 Call `NWGetDriveStatus` (page 614) to determine if the specified drive is available as a network drive.
- 3 Call `NWParsePath` (page 624) to determine if a connection exists to the server specified in the path.
- 4 If a connection does not exist to the specified server, establish a connection.
- 5 Remove the server name from the path by calling `NWStripServerOffPath` (page 631).
- 6 Map the drive by calling `NWSetDriveBase` (page 627).

22.2.1 Mapping a Network Drive Example

NOTE: taken from SETDRIVE.C in the \EXAMPLES directory

```
/
*****
SETDRIVE.C
*****
```

SETDRIVE.C demonstrates how to map a drive to a NetWare server using `NWSetDriveBase()`.

USAGE: SETDRIVE <drive number> <server name> <path>

drive number: 1=A, 2=B, etc.
server name : name of the server
path : path to map, including volume name.

```
Exmpl:  SETDRIVE 10 MYSERVER SYS:\DIRECTORY\SUBDIR
```

```
(maps drive J to MYSERVER\SYS:\DIRECTORY\SUBDIR)
```

```
*****/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <nwclxcon.h>
#include <nwcalls.h>
#include <nwnet.h>
#include <nwlocale.h>

int main(int argc, char *argv[])
{
    NWCONN_HANDLE      connHandle;
    NWCCODE             ccode;
    LCONV              lconvInfo;
    NWDSContextHandle  dContext;
    nbool8             bbDoLogout = N_FALSE;
    nstr8              strUserName[NW_MAX_USER_NAME_LEN];
    nstr8              strUserPassword[50];
    nstr8              strServerName[NW_MAX_SERVER_NAME_LEN];
    nstr8              strDriveNumber[3];
    nstr8              strPath[50];
    nstr8              strContext[MAX_DN_CHARS + 1];

    if(argc != 4)
    {
        printf ("\nUsage:  SETDRIVE <drivenumber> <server>
                <directory>\n");
        printf ("  drivenumber : A=1, B=2, etc...\n");
        printf ("  server       : name of server \n");
        printf ("  directory    : path of directory, including
                volume\n");
        printf ("\nExample: SETDRIVE 10 myserver
                sys:\\users\\mydir\\mysubdir\n");
        exit(1);
    }

    strcpy(strDriveNumber, strupr(argv[1]));
    strcpy(strServerName, strupr(argv[2]));
    strcpy(strPath, strupr(argv[3]));

    /* Initialize libraries */
    ccode = NWCallsInit(NULL, NULL);
    if(ccode)
    {
        printf("\nNWCallsInit returned %04X", ccode);
        exit(1);
    }

    NWLlocaleconv(&lconvInfo);
```

```

ccode = NWInitUnicodeTables(lconvInfo.country_id,
    lconvInfo.code_page);
if(ccode)
{
    printf("NWInitUnicodeTables() returned: %04X\n", ccode);
    exit(1);
}

/* Create a context and authenticate to NDS if necessary */
ccode = NWDSCreateContextHandle(&dContext);
if(ccode)
{
    printf("NWDSCreateContextHandle returned: %04lX\n", ccode);
    goto _FreeUnicodeTables;
}

ccode = NWDSGetContext(dContext, DCK_NAME_CONTEXT, strContext);
if(ccode)
{
    printf("\nNWDSGetContext returned %04X", ccode);
    exit(1);
}
printf("\nstrContext:  %s", strContext);

/* Must authenticate if not already authenticated to NDS */
if(!NWIsDSAuthenticated())
{
    printf("\nMust authenticate to NDS");
    printf("\nEnter User Name: ");
    gets(strUserName);
    printf("Enter User Password: ");
    gets(strUserPassword);

    ccode = NWDSLogin(dContext, 0, strUserName, strUserPassword, 0);
    if(ccode)
    {
        printf("\nNWDSLogin returned %X", ccode);
        goto _FreeContext;
    }
    else
        bbDoLogout = N_TRUE;
}

/* Open a connection to the specified server */

printf("\nstrServerName: %s", strServerName);
ccode = NWCCOpenConnByName(
    /* start Conn Handle */ 0,
    /* name                  */ strServerName,
    /* name format          */ NWCC_NAME_FORMAT_BIND,
    /* open state           */ NWCC_OPEN_UNLICENSED,
    /* tran type            */ NWCC_TRAN_TYPE_IPX,
    /* Connection Handle    */ &connHandle);

```

```

if(ccode)
{
    printf( "\nNWCCOpenConnByName returned %04x", ccode );
    goto _Logout;
}

ccode = NWSetDriveBase(
    /* drive number      */ (nuint16)atoi(strDriveNumber),
    /* handle to server */ connHandle,
    /* directory handle */ 0,
    /* directory path   */ strPath,
    /* reserved         */ 0);

if(ccode)
{
    printf("\nNWSetDriveBase returned %04X\n", ccode);
    goto _FreeConnection;
}

/* Successful termination */
return(0);

/* Unsuccessful termination */
_FreeConnection:
    NWCCCloseConn(connHandle);

_Logout:
    if (bbDoLogout == N_TRUE)
        NWDSLogout(dContext);

_FreeContext:
    NWDSFreeContext(dContext);

_FreeUnicodeTables:
    NWFreeUnicodeTables();

return(1);
}

```

Path and Drive Functions

23

This documentation alphabetically lists the Path and Drive functions and describes their purpose, syntax, parameters, and return values.

- ♦ “ConvertNameToFullPath” on page 606
- ♦ “ConvertNameToVolumePath” on page 607
- ♦ “NWDeleteDriveBase” on page 608
- ♦ “NWGetDirBaseFromPath” on page 610
- ♦ “NWGetDriveInformation” on page 612
- ♦ “NWGetDriveStatus” on page 614
- ♦ “NWGetDriveStatusConnRef” on page 616
- ♦ “NWGetFirstDrive” on page 618
- ♦ “NWGetPathFromDirectoryBase” on page 620
- ♦ “NWParseNetWarePath” on page 622
- ♦ “NWParsePath” on page 624
- ♦ “NWSetDriveBase” on page 627
- ♦ “NWSetInitDrive (obsolete 7/99)” on page 629
- ♦ “NWStripServerOffPath” on page 631
- ♦ “ParsePath” on page 632
- ♦ “SetWildcardTranslationMode” on page 634
- ♦ “StripFileServerFromPath” on page 635

ConvertNameToFullPath

Converts a path to an absolute path specification that includes a volume specification

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Path and Drive

Syntax

```
#include <stdlib.h>
#include <nwdir.h>

int ConvertNameToFullPath (
    char *partialPath,
    char *fullPath);
```

Parameters

partialPath

(IN) Points to a string containing the partial path that is to be converted to a complete path.

fullPath

(OUT) Points to the buffer where the complete path is to be returned (maximum 255 characters).

Return Values

0 (0x00)	ESUCCESS: Only fails if the <code>partialPath</code> parameter is not valid.
22 (0x16)	EBADHNDL

Remarks

`ConvertNameToFullPath` accepts a file name, or any relative or absolute path, and returns the absolute path (including a volume specification).

Call `ConvertNameToFullPath` when a user is entering a file name (which may or may not be entered as a full path specification) and you want a full path specification to open the file.

`ConvertNameToFullPath` uses `ParsePath` to construct the `fullPath` parameter string.

See Also

[ConvertNameToVolumePath \(page 607\)](#), [ParsePath \(page 632\)](#)

ConvertNameToVolumePath

Converts a path to an absolute path specification that does not include the volume specification

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Path and Drive

Syntax

```
#include <nwdir.h>

int ConvertNameToVolumePath (
    char *fileName,
    char *path);
```

Parameters

fileName

(IN) Points to the name of the file that is to be converted to a complete path from the volume.

path

(OUT) Points to the buffer where the complete path is to be returned (maximum 255 characters).

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
22	(0x16)	EBADHNDL

Remarks

ConvertNameToVolumePath accepts a filename, or any relative or absolute path, and returns the absolute path (not including a volume specification). The volume name is not included in the path.

Call ConvertNameToVolumePath when a user is entering a filename (which may or may not be entered as a full path specification) and you want a full path specification to open the file.

See Also

[ConvertNameToFullPath \(page 606\)](#)

NWDeleteDriveBase

Deletes a network drive mapping

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT*, Windows* 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWDeleteDriveBase (
    nuint16    driveNum,
    nuint16    driveScope);
```

Delphi Syntax

```
uses calwin32

Function NWDeleteDriveBase
  (driveNum : nuint16;
   driveScope : nuint16
  ) : NWCCODE;
```

Parameters

driveNum

(IN) Specifies the drive number whose mapping is being deleted (A=1, B=2, ...).

driveScope

Reserved for Novell® use only; must be 0.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8804	BAD_DRIVE_BASE
0x8836	INVALID_PARAMETER
0x883C	NOT_MY_RESOURCE
0x8875	INVALID_DRIVE_NUM

Remarks

If `driveNum` is zero, the current drive will be deleted if it belongs to the NetWare® OS.

Most operating systems will determine if the path is valid before `NWDeleteDriveBase` returns.

Under Windows 95 and Windows 98, `0x0003 Path Not Found` will be returned if the path is invalid.

Under Windows NT, `INVALID_PARAMETER` will be returned if an unmapped drive is being referenced. `INVALID_DRIVE_NUM` will be returned if an invalid drive number is being used.

Under NLM, `INVALID_SHELL_CALL` is always returned.

See Also

[NWSetDriveBase \(page 627\)](#)

NWGetDirBaseFromPath

Gets a volume number, a directory base for the specified name space, and a directory base for the DOS name space entry

Local Servers: blocking

Remote Servers: blocking

Platform: NLM

NetWare Server: 3.12, 3.2, 4.x, 5.x, 6.x

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Name Space

Syntax

```
#include <nwfileng.h>

N_EXTERN_LIBRARY( NWCCODE )NWGetDirBaseFromPath (
    char    *path,
    BYTE    nameSpace,
    LONG    *volNum,
    LONG    *NSDirBase,
    LONG    *DOSDirBase);
```

Parameters

path

(IN) Points to the directory path to generate a directory base (number) for.

nameSpace

(IN) Specifies the name space to generate the directory base (number) for.

volNum

(OUT) Points to the volume number that corresponds with `path`.

NSDirBase

(OUT) Points to a directory index for the specified name space.

DOSDirBase

(OUT) Points to a directory index for the DOS name space of the entry.

Return Values

If `NWGetDirBaseFromPath` succeeds, it returns zero. Otherwise, it returns a nonzero error code.

Remarks

NWGetDirBaseFromPath gets a volume number, a directory base for the specified name space, and a directory base for the DOS name space for the entry.

NWGetDriveInformation

Returns information about the specified drive

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWGetDriveInformation (
    nuint16          driveNum,
    nuint16          mode,
    NWCONN_HANDLE N_FAR *conn,
    NWDIR_HANDLE N_FAR *dirHandle,
    pnuint16         driveScope,
    pnstr8           dirPath);
```

Delphi Syntax

```
uses calwin32

Function NWGetDriveInformation
  (driveNum : nuint16;
   mode : nuint16;
   Var conn : NWCONN_HANDLE;
   Var dirHandle : NWDIR_HANDLE;
   driveScope : pnuint16;
   dirPath : pnstr8
  ) : NWCCODE;
```

Parameters

driveNum

(IN) Specifies the drive number for which to get the status (A=1, B=2, C=3, . . .); pass 0 for current drive.

mode

Currently unused.

conn

(OUT) Points to the connection ID of the server the drive is currently mapped to.

dirHandle

(OUT) Points to the directory handle associated with the specified drive.

driveScope

(OUT) Points to the drive scope (currently returns GLOBAL).

dirPath

(OUT) Points to the current directory of the specified drive.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x000F	DOS_INVALID_DRIVE
0x883C	NOT_MY_RESOURCE
0x89FF	INVALID_DRIVE_NUMBER

Remarks

If `driveNum` is 0, information about the current drive is returned.

`DOS_INVALID_DRIVE` is returned if the drive is not defined.

If VLMS are running, `dirHandle` returns 0. VLMS do not associate a directory handle with a mapped drive, no directory handle can be returned. For example, if NETX version 3.32 is running, `NWGetDriveInformation` will return a valid `dirHandle` (non-zero) and a valid `dirPath`. If VLM version 1.20 is running, `NWGetDriveInformation` returns a `dirHandle` of zero and a valid `dirPath` (the same `dirPath` returned when NETX was running).

Under Windows NT, a `dirHandle` will not be returned. Under all other platforms, if `dirHandle` does not point to NULL, a `dirHandle` will be returned if NETX support is available. Otherwise, `NWGetDriveInformation` will return `NWE_REQUESTER_FAILURE` (0x88FF).

Under NLM, `INVALID_SHELL_CALL` is always returned.

See Also

[NWGetFirstDrive \(page 618\)](#)

NWGetDriveStatus

Returns the status of the specified drive and, optionally, the associated connection and its path in various formats

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWGetDriveStatus (
    nuint16          driveNum,
    nuint16          pathFormat,
    pnuint16         status,
    NWCONN_HANDLE N_FAR *conn,
    pustr8           rootPath,
    pustr8           relPath,
    pustr8           fullPath);
```

Delphi Syntax

```
uses calwin32

Function NWGetDriveStatus
  (driveNum : nuint16;
   pathFormat : nuint16;
   status : pnuint16;
   Var conn : NWCONN_HANDLE;
   rootPath : pustr8;
   relPath : pustr8;
   fullPath : pustr8
  ) : NWCCODE;
```

Parameters

driveNum

(IN) Specifies the drive number for which to get the status (A=1, B=2, C=3, . . .); pass 0 for current drive.

pathFormat

(IN) Specifies the desired format for the return paths.

status

(OUT) Points to a bit mask indicating if the drive is local and/or networked.

conn

(OUT) Points to the connection handle of the path `driveNum` is mapped to, if any (optional).

rootPath

(OUT) Points to the base path `driveNum` is mapped to (optional).

relPath

(OUT) Points to the path (relative to the `rootPath` parameter) to which the drive number is mapped (optional).

fullPath

(OUT) Points to the full path of `driveNum`, if it is a network drive (optional).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x000F	NW_INVALID_DRIVE
0x8800	Unknown Error Occurred; Unable to Complete Request
0x883C	NOT_MY_RESOURCE

Remarks

Currently, `NWGetDriveStatus` returns the status of local drives, but does not return path strings for these paths to prevent critical errors from occurring on removable drives. (May change with future releases.)

See [Section 21.1, "Path Parameters," on page 599](#) for input values and examples of returned information.

`NW_LOCAL_DRIVE` indicates the specified drive letter is lower than the first networked drive which usually defaults to F: and is set in the `net.cfg` file.

See Also

[NWGetFirstDrive \(page 618\)](#)

NWGetDriveStatusConnRef

Returns the status of the specified drive and, optionally, the associated connection reference and its path in various formats

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY (NWCCODE) NWGetDriveStatusConnRef (
    nuint16      driveNum,
    nuint16      pathFormat,
    pnuint16     status,
    pnuint32     connRef,
    pnstr8       rootPath,
    pnstr8       relPath,
    pnstr8       fullPath);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWGetDriveStatusConnRef (
    driveNum : nuint16;
    pathFormat : nuint16;
    status : pnuint16;
    connRef : pnuint32;
    rootPath : pnstr8;
    relPath : pnstr8;
    fullPath : pnstr8
) : NWCCODE;
```

Parameters

driveNum

(IN) Specifies the drive number for which to return the status (A=1, B=2, C=3, ...). Pass 0 for the current drive.

pathFormat

(IN) Specifies the desired format for the return paths.

status

(OUT) Points to a bit mask indicating if the drive is local and/or networked.

connRef

(OUT) Points to the connection reference of the specified drive (optional).

rootPath

(OUT) Points to the base path to which the specified drive is mapped (optional).

relPath

(OUT) Points to the path (relative to the `rootPath` parameter) to which the drive number is mapped (optional).

fullPath

(OUT) Points to the full path of the specified drive if it is a network drive (optional).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x000F	NW_INVALID_DRIVE
0x8836	INVALID_PARAMETER

Remarks

NWGetDriveStatusConnRef does not work with local drives.

See [Section 21.1, “Path Parameters,” on page 599](#) for input values and examples of returned information.

NW_LOCAL_DRIVE indicates the specified drive letter is lower than the first networked drive which usually defaults to F: and is set in the `net.cfg` file.

Under NLM, INVALID_SHELL_CALL is always returned.

See Also

[NWCCGetPrimConnRef, NWGetDriveStatus \(page 614\)](#)

NWGetFirstDrive

Returns the first non-local drive

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY( NWCCODE )NWGetFirstDrive (
    puint16 firstDrive);
```

Delphi Syntax

```
uses calwin32

Function NWGetFirstDrive
    (firstDrive : puint16
) : NWCCODE;
```

Parameters

firstDrive

(OUT) Points to the first non-local drive (A=1, B=2, C=3. . .).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x000F	Unknown error occurred

Remarks

If an unknown error occurs while obtaining drive information, NWGetFirstDrive returns 0x000F; this is very rare.

Under NLM, INVALID_SHELL_CALL is always returned.

See Also

[NWGetDriveStatus \(page 614\)](#)

NWGetPathFromDirectoryBase

Returns the path name from an entry in the directory entry table for a NetWare server

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY (NWCCODE) NWGetPathFromDirectoryBase (
    NWCONN_HANDLE   conn,
    nuint8           volNum,
    nuint32          dirBase,
    nuint8           namSpc,
    pnuint8          len,
    pnstr8           pathName);
```

Delphi Syntax

```
uses calwin32

Function NWGetPathFromDirectoryBase
  (conn : NWCONN_HANDLE;
   volNum : nuint8;
   dirBase : nuint32;
   namSpc : nuint8
   len : pnuint8;
   pathName : pnstr8
  ) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNum

(IN) Specifies the volume number.

dirBase

(IN) Specifies the directory entry number in the name space specified by the `namSpc` parameter.

namSpc

(IN) Specifies the name space used by the directory entry number (see [Section 20.5, “Name Space Flag Values,”](#) on page 595).

len

(OUT) Points to the path length and specifies how much of the buffer pointed to by the `pathName` parameter was used (initialize to the length of the buffer to hold the path).

pathName

(OUT) Points to the buffer containing the path name (maximum 255 characters).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x899C	INVALID_PATH

Remarks

`NWGetPathFromDirectoryBase` maps a directory entry number to a path under a specified name space. The path is returned as a group of components. Each directory, subdirectory, or file in the path is considered to be a component. Each component is length preceeded and followed by the next component.

For example, `pathName` returns the `users/jdoe/working` directory returned as:

```
5users4jdoe6working
```

You must allocate memory for the buffer pointed to by the `pathName` parameter.

`NWGetPathFromDirectoryBase` returns the path in the `pathName` parameter as a length-preceded array with generic separators.

NCP Calls

0x2222 23 243 Map Directory Number to Path

NWParseNetWarePath

Parses a path and returns the connection handle, directory handle, and new path to be used by subsequent NetWare requests

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(NWCCODE) NWParseNetWarePath (
    const nstr8 N_FAR    *path,
    NWCONN_HANDLE N_FAR *conn,
    NWDIR_HANDLE N_FAR  *dirHandle,
    pnstr8                newPath);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWParseNetWarePath
  (const path : pnstr8;
  Var conn : NWCONN_HANDLE;
  Var dirHandle : NWDIR_HANDLE;
  newPath : pnstr8
  ) : NWCCODE;
```

Parameters

path

(IN) Points to the path (in capital letters) being parsed.

conn

(OUT) Points to the NetWare server connection handle.

dirHandle

(OUT) Points to the directory handle.

newPath

(OUT) Points to the new path, relative to the directory handle—this parameter should be a buffer of at least 256 characters.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x880F	NO_CONNECTION_TO_SERVER
0x883C	NOT_MY_RESOURCE

Remarks

NWParseNetWarePath does not check the validity of any volume or directory names in the path string.

path must be specified in capital letters or the call to NWParseNetWarePath fails.

If the path to be parsed is relative to the current directory, NWParseNetWarePath assumes the current drive and returns a complete path on all platforms. If the path is on a local drive, NWParseNetWarePath returns NOT_MY_RESOURCE. If the path specifies a NetWare server name and there are no connections to that NetWare server, NWParseNetWarePath returns NO_CONNECTION_TO_SERVER.

Under all platforms, NWParseNetWarePath returns zero (0) in dirHandle and a full path (volume:path) in newPath.

NCP Calls

0x2222 23 17 Get File Server Information
0x2222 23 22 Get Station's Logged Info (old)
0x2222 23 28 Get Station's Logged Info
0x2222 104 1 Ping for NDS NCP

See Also

[NWParsePath \(page 624\)](#)

NWParsePath

Parses a path string

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWParsePath (
    constr nstr8 N_FAR    *path,
    pnstr8                serverName,
    NWCONN_HANDLE N_FAR  *conn,
    pnstr8                volName,
    pnstr8                dirPath);
```

Delphi Syntax

```
uses calwin32

Function NWParsePath
  (const path : pnstr8;
   serverName : pnstr8;
   Var conn : NWCONN_HANDLE;
   volName : pnstr8;
   dirPath : pnstr8
  ) : NWCCODE;
```

Parameters

path

(IN) Points to the path to be parsed.

serverName

(OUT) Points to the server name (48 characters, optional).

conn

(OUT) Points to the connection handle of the server (optional).

volName

(OUT) Points to the volume name (17 characters, optional).

dirPath

(OUT) Points to the directory portion of the path; this parameter should be a buffer of at least 256 characters.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x880F	NO_CONNECTION_TO_SERVER

Remarks

If `conn` is not NULL, a new connection handle will be returned by `NWParsePath`. You will need to ensure this connection handle is properly closed.

If the path to be parsed is relative to the current directory, `NWParsePath` assumes the current drive and path so a complete path specification is returned.

```
IF k: is the current drive
AND \dir1 is the current directory on k:
AND dir2 is a directory in dir1
THEN calling NWParsePath with path pointing to "dir2" will cause
dirPath to return "dir1\dir2".
```

If the path to be parsed contains a map rooted drive, `dirPath` will be set to the complete directory path from the volume level.

```
IF k:is map rooted to server1/sys:dir1\
AND dir2 is a directory in dir1
THEN calling NWParsePath with path pointing to "k:dir2" will cause
dirPath to return "dir1\dir2" even though the DOS path is k:\dir2.
```

If the path to be parsed is relative to the current directory, the entire directory path will be returned, *without* a preceding backslash character.

```
IF k: is mapped to server1/sys:
AND the current directory path for k: is dir1
AND dir2 is a directory in dir1
THEN calling NWParsePath with path pointing to "k:dir2" will cause
dirPath to return "dir1\dir2".
```

If the path to be parsed is on the root directory, `dirPath` will return with a preceding backslash character even if one is not included in the call. This is the only case that will return a preceding backslash character.

```
IF k: is mapped to server1/sys:
AND the current directory path on k: is the root
AND dir1 is a directory on the root
THEN calling NWParsePath with path pointing to "k:dir1" will cause
dirPath to return "\dir1". Note the preceding backslash character in this
case. This is the same for local drives and mapped drives.
```

serverName, conn, volName, and dirPath are optional. Substitute NULL if no returns are desired. However, all parameter positions must be filled.

If the path is on a local drive, return information is placed in the return parameters as follows:

```
serverName  zero-length string
conn        0
volName     drive letter
dirPath     directories from drive letter
```

NWParsePath does not guarantee the path actually exists.

If the path specifies a NetWare server name and there are no connections to that NetWare server, NO_CONNECTION_TO_SERVER is returned. The path specification can be any of the following:

Specification	Function
drive:path	Drive letter is used to determine the network information, if any.
vol:path	Volume and path will be assumed to be relative to the default server.
server vol:path	Information is copied to the associated return buffers and, if requested, the connection handle is obtained using the server name.
path	Current drive is used to determine all the information.

If a map rooted drive is used, dirPath will be set to the complete directory path from the volume level.

NCP Calls

```
0x2222 23 17 Get File Server Information
0x2222 23 22 Get Station's Logged Info (old)
0x2222 23 28 Get Station's Logged Info
0x2222 104 1 Ping for NDS NCP
```

See Also

[NWParseNetWarePath \(page 622\)](#)

NWSetDriveBase

Maps the target drive to the specified directory path

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY(NWCCODE) NWSetDriveBase (
    nuint16          driveNum,
    NWCONN_HANDLE   conn,
    NWDIR_HANDLE     dirHandle,
    const nstr8 N_FAR *dirPath,
    nuint16          driveScope);
```

Delphi Syntax

```
uses calwin32;

Function NWSetDriveBase
  (driveNum : nuint16;
   conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   dirPath : pnstr8;
   driveScope : nuint16
  ) : NWCCODE;
```

Parameters

driveNum

(IN) Specifies the drive number of the drive being mapped (0=current, 1=A, 2=B, . . .).

conn

(IN) Specifies the NetWare server connection handle to which the drive is mapped.

dirHandle

(IN) Specifies the directory handle associated with `dirPath`.

dirPath

(IN) Points to the directory path the drive will be mapped to. `dirPath` is relative to `dirHandle`, unless `dirHandle` is 0.

driveScope

Reserved for Novell use only; must be 0.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8802	DRIVE_IN_USE (Windows NT): The drive number is already mapped
0x8803	DRIVE_CANNOT_MAP
0x883C	NOT_MY_RESOURCE: Trying to map a local drive
0x8875	INVALID_DRIVE_NUM
0x8998	VOLUME_DOES_NOT_EXIST
0x899B	BAD_DIRECTORY_HANDLE
0x899C	INVALID_PATH
0x89FF	INVALID_DRIVE_NUMBER (Windows NT): An invalid drive number is being used

Remarks

If the specified drive number is zero, the current drive will be remapped to the specified path. For other drive numbers, if the target drive is already mapped, the mapping must be deleted by calling `NWDeleteDriveBase` before calling `NWSetDriveBase`.

Under all platforms, CD-ROM drives cannot be mapped.

The server name should not be specified in the `dirPath` parameter. Specify the server name in the `conn` parameter. Under `NETX.EXE`, the server name can be parsed, but `VLMs` do not parse out the server name.

Under `NLM`, `INVALID_SHELL_CALL` is always returned.

See Also

[NWDeleteDriveBase \(page 608\)](#), [NWGetDriveStatus \(page 614\)](#)

NWSetInitDrive (obsolete 7/99)

Sets the initial drive on the specified NetWare server but is now obsolete.

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>

N_EXTERN_LIBRARY (NWCCODE) NWSetInitDrive (
    NWCONN_HANDLE conn);
```

Delphi Syntax

```
uses calwin32;

Function NWSetInitDrive
    (conn : NWCONN_HANDLE
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle on which to set the initial drive.

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION

Remarks

NWSetInitDrive (obsolete 7/99) is used under OS/2 to set the mapping for drive L, the OS/2 drive containing the system login for attaching to a server.

NWSetInitDrive (obsolete 7/99) can be called from all platforms; however, it will only set the correct drive mapping under OS/2. When called from all other platforms, NWSetInitDrive (obsolete 7/99) returns SUCCESSFUL without setting the correct drive mapping.

Under NLM, INVALID_SHELL_CALL is always returned.

NWStripServerOffPath

Parses a server or volume path, copies the server name to the buffer specified by server, and returns a pointer to the volume path

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Path and Drive

Syntax

```
#include <nwdpath.h>
or
#include <nwcalls.h>
```

```
N_EXTERN_LIBRARY(pnstr8) NWStripServerOffPath (
    constr nstr8 N_FAR *path,
    pnstr8          server);
```

Delphi Syntax

```
uses calwin32
```

```
Function NWStripServerOffPath
    (path : pnstr8;
     server : pnstr8
    ) : pnstr8;
```

Parameters

path

(IN) Points to a string containing a server volume path.

server

(OUT) Points to a 48-character buffer for the server name (optional).

Return Values

These are common return values; see [Return Values \(Return Values for C\)](#) for more information.

0x0000	path passed in was NULL
character pointer	pointer to the volume path

See Also

[NWParsePath \(page 624\)](#), [NWParseNetWarePath \(page 622\)](#)

ParsePath

Separates a full path into server, volume, and directory specifications

Local Servers: nonblocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Path and Drive

Syntax

```
#include <stdlib.h>
#include <nwdir.h>

int ParsePath (
    char *path,
    char *server,
    char *volume,
    char *directories);
```

Parameters

path

(IN) Points to the string containing the path to be parsed and can include a server name (255 character maximum).

server

(OUT) Points to the buffer in which to return the server name (48 character maximum).

volume

(OUT) Points to the buffer in which to return the volume name (16 character maximum).

directories

(OUT) Points to the buffer in which to return the directory specification (255 character maximum).

Return Values

Value	Hex	Constant and Definition
0	(0x00)	ESUCCESS: Fails if an invalid path is passed.
22	(0x16)	EBADHNDL

Remarks

ParsePath parses the given path and separates it into server, volume, and directory specifications. Even if the path is not complete (or it is relative to the current working directory), ParsePath returns the complete path specification.

Strings for the `server`, `volume`, and `directories` parameters are always converted to uppercase characters.

See Also

[StripFileServerFromPath \(page 635\)](#)

SetWildcardTranslationMode

Specifies whether wildcard translation is to take place when parsing pathnames and filenames

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Path and Drive

Syntax

```
#include <nwdir.h>
```

```
BYTE SetWildcardTranslationMode (
    BYTE newMode);
```

Parameters

newMode

(IN) Specifies the new translation mode (TRUE or FALSE).

Return Values

Returns the old translation mode.

Remarks

SetWildcardTranslationMode enables (TRUE) or disables (FALSE) translation of the following wildcards when parsing path and filenames:

* asterisk

? question mark

. period

When translation is enabled, the high-order bit is changed for all wildcard characters that are parsed in any subsequent file or directory service function. If the high-order bit is 0, it is set to a value of 1. If the high-order bit is 1, it is set to 0.

NetWare uses its own set of rules to interpret wildcards in pathnames. If the high-order bit of a wildcard character is a 1, NetWare interprets that character as a DOS wildcard (this is called an augmented wildcard) and uses DOS rules for interpretation of that wildcard.

StripFileServerFromPath

Removes the name of the server from a full path specification

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Path and Drive

Syntax

```
#include <stdlib.h>
#include <nwdir.h>

char * StripFileServerFromPath (
    char *path,
    char *server);
```

Parameters

path

(IN) Points to the string containing the path from which to remove the server name.

server

(OUT) Points to the buffer in which to place the stripped server name (48 character maximum).

Return Values

Returns a pointer to a path specification stripped of the server name.

Remarks

StripFileServerFromPath removes the name of the server from a path specification. If the `path` parameter does not include a server specification, StripFileServerFromPath returns the original path. If the `path` parameter does include a server specification, the returned value begins with the volume specification.

See Also

[ParsePath \(page 632\)](#)

Server-Based Data Migration Concepts

24

This documentation describes Server-Based Data Migration, its functions, and features.

NOTE: Writing a data migrator is a time-consuming project. Therefore, Server-Based Data Migration is not designed for actually writing a migrator but for writing an NLM application that uses a migrator that Novell or another party has already written. If you are interested in writing a migrator, Novell Developer Relations can provide you with help and resources.

Data Migration Services give system administrators the ability to migrate (move) files from primary storage to secondary (slower) storage. The migrated files appear to the Supervisor to be located on primary storage; the directory structure is kept intact. When the Supervisor or user accesses a migrated file, it is de-migrated in real time to primary storage for the user.

Some examples of secondary storage are optical jukeboxes, DAT jukeboxes, and so forth. Novell provides a device driver for the HP 5 1/4" Optical Jukebox.

24.1 Advantages of Data Migration Applications

Because the NetWare® file system continues to display the files as if they were still resident on the volume, users can migrate or de-migrate files at will. In addition, there are data migration functions for automatic dynamic migration and de-migration.

If your NLM is a database that could grow very large, your users can benefit from being able to migrate it when the appropriate time comes.

There is no limit to the amount of files users can migrate. Thus, a relatively small NetWare volume—for example, one on a 100-megabyte internal hard disk—becomes a larger virtual storage area when certain strategic files have been migrated.

CD ROMs or disk subsystems can hold huge quantities of data at the ready, so a data migration application can optimize a network's utilization of the available storage space. For example, images (graphics) lend themselves to being migrated because they are large files that typically are seldom accessed. Similarly, databases can grow to large proportions and might be migratable under certain conditions.

All things being equal (file size, file type, and so forth), a file that has been migrated to a CD ROM or disk subsystem can be retrieved in almost the same time as it would take to retrieve it from the NetWare volume.

Novell is providing users with three software modules that allow them to do real-time data migration:

- ♦ **High Capacity Storage Subsystem (HCSS):** A front-end data migration application that allows NetWare 4.x administrators to migrate data based upon a high and low water mark. The administrator sets a high and low percentage mark that indicates when HCSS should migrate files based on the last accessed-date. Each day HCSS migrates files to the low water mark specification. Any time primary storage reaches the high water mark, HCSS dynamically

begins to migrate files to secondary storage. You could write a data migration NLM to migrate files any time a different set of conditions exists, depending on your users needs.

- ◆ Two support-module NLM applications: Up to 32 support modules can be written to register up to 32 different types of storage devices—hard disk, tape, CD ROM—with NetWare data migration NLM, the DM. Novell supplies two sample support modules with NetWare 4.x, one for the HP 5 1/4" Optical Jukebox CD ROM and one for the hard disk.

For help writing a support module, contact Novell Developer Relations.

24.2 Server-Based Data Migration Functions

These are the server-based data migration functions and their purposes:

NWMoveFileToDM	Migrate a file
NWMoveFileFromDM	De-migrate a file
NWPeekFileData	Read part of a migrated file
NWSetDefaultSupportModule	Change the default support modules
NWGetDataMigratorInfo	Get version number of DM and total number of accompanying support modules
NWGetDefaultSupportModule	Get the default read/write support module ID
NWGetDMFileInfo	Get file information on the DM (path, name space, and so forth)
NWGetDMVolumeInfo	Get volume information (total number of files that have been migrated to a certain volume and their total size)
NWGetSupportModuleInfo	Determine which support modules are currently registered
NWIsDataMigrationAllowed	Determine if data migration is allowed on a particular volume

Server-Based Data Migration Functions

25

This documentation alphabetically lists the Server-Based Data Migration functions and describes their purpose, syntax, parameters, and return values.

- ◆ “NWGetDataMigratorInfo” on page 640
- ◆ “NWGetDefaultSupportModule” on page 641
- ◆ “NWGetDMFileInfo” on page 642
- ◆ “NWGetDMVolumeInfo” on page 644
- ◆ “NWGetSupportModuleInfo” on page 645
- ◆ “NWIsDataMigrationAllowed” on page 647
- ◆ “NWMoveFileFromDM” on page 648
- ◆ “NWMoveFileToDM” on page 649
- ◆ “NWPeekFileData” on page 650
- ◆ “NWSetDefaultSupportModule” on page 652

For cross-platform functionality, see [Developing NLMs with Cross-Platform Functions \(NDK: NLM Development Concepts, Tools, and Functions\)](#) and call the alternative function listed with each NLM function.

NWGetDataMigratorInfo

Obtains information about a data migration NLM application

Local Servers: blocking

Remote Servers: blocking

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>

void NWGetDataMigratorInfo (
    LONG    *DMPresentFlag,
    LONG    *majorVersion,
    LONG    *minorVersion,
    LONG    *numberOfSupportModules);
```

Parameters

DMPresentFlag

(OUT) Receives the status of the data migration NLM.

majorVersion

(OUT) Receives the major version number of the data migration NLM.

minorVersion

(OUT) Receives the minor version number of the data migration NLM.

numberOfSupportModules

(OUT) Receives the number of modules supported by the data migration NLM.

Remarks

For cross-platform functionality, call [NWGetDataMigratorInfo \(page 20\)](#).

This function obtains the following information about the data migration NLM:

- ◆ Whether it is loaded and running
- ◆ Its major and minor version numbers
- ◆ The number of modules supported by the NLM

The `DMPresentFlag` receives -1 if the data migration NLM is loaded and running. If `DMPresentFlag` receives 0, the data migration NLM is not loaded.

NWGetDefaultSupportModule

Obtains the default read/write support module ID

Local Servers: blocking

Remote Servers: blocking

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>

LONG NWGetDefaultSupportModule (
    LONG *defaultSupportModuleID);
```

Parameters

defaultSupportModuleID

(OUT) Receives the ID number of the default support module.

Return Values

0	Successful.
---	-------------

Remarks

For cross-platform functionality, call [NWSetDefaultSupportModule \(page 37\)](#).

See Also

[NWSetDefaultSupportModule \(page 652\)](#)

NWGetDMFileInfo

Obtains information about a file that has been migrated to long-term storage

Local Servers: blocking

Remote Servers: blocking

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>
```

```
LONG NWGetDMFileInfo (  
    char    *path,  
    LONG    nameSpace,  
    LONG    *supportModuleID,  
    LONG    *validDataStreams,  
    BYTE    *estRetrievalTime,  
    LONG    *info);
```

Parameters

path

(IN) Points to the - path of a file.

nameSpace

(IN) Specifies the name space of the path.

supportModuleID

(OUT) Receives the assigned ID number of the support module that migrated the data to long-term storage.

validDataStreams

(OUT) Receives the data streams that are supported by the data migrator.

estRetrievalTime

(OUT) Receives an estimate of how long data retrieval will take.

info

(OUT) Points to more file information.

Return Values

0	Successful.
---	-------------

Remarks

For cross-platform functionality, call [NWGetDMFileInfo](#) (page 24).

NWGetDMVolumeInfo

Obtains information about the volume from which data has been migrated to long-term storage

Local Servers: blocking

Remote Servers: blocking

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>

LONG NWGetDMVolumeInfo (
    LONG    volume,
    LONG    supportModuleID,
    LONG    *numberOfFilesMigrated,
    LONG    *totalMigratedSize,
    LONG    *spaceUsed,
    LONG    *limboUsed,
    LONG    *spaceMigrated,
    LONG    *filesLimbo);
```

Parameters

volume

(IN) Specifies the volume that contains migrated files.

numberOfFilesMigrated

(OUT) Receives the number of files on the volume that have been migrated to long-term storage.

totalMigratedSize

(OUT) Receives the total size needed to recover all data migrated on the volume.

Return Values

0	Successful.
---	-------------

Remarks

For cross-platform functionality, call [NWGetDMVolumeInfo \(page 27\)](#).

NWGetSupportModuleInfo

Obtains information about data migration support modules

Local Servers: blocking

Remote Servers: blocking

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>

LONG NWGetSupportModuleInfo (
    LONG    informationLevel,
    LONG    supportModuleID,
    void    *returnInfo,
    LONG    *returnInfoLen);
```

Parameters

informationLevel

(IN) Specifies the type of information requested.

supportModuleID

(IN) Specifies the data migration support module to return information for.

returnInfo

(OUT) Points to the area where the information from this function is stored.

returnInfoLen

(OUT) Receives the length of the information returned.

Return Values

0	Successful.
---	-------------

Remarks

For cross-platform functionality, call [NWGetSupportModuleInfo \(page 30\)](#).

The type of information that this function returns depends on the value specified in `informationLevel`. The following indicates the type of information returned for each value of `informationLevel`:

0	NWGetSupportModuleInfo returns information about the data migration support module in the <code>returnInfo</code> parameter.
1	NWGetSupportModuleInfo returns a list of all loaded data migration support module ID numbers in the <code>returnInfo</code> parameter.

The `returnInfo` parameter receives a different type of structure depending on the type of information requested. If information about a particular data migration support module is requested, `returnInfo` receives a structure of type `SUPPORT_MODULE_INFO`, which is defined in `\nlm\nit\nwdatamg.h` as follows:

```
typedef struct {
    LONG    IOStatus;
    LONG    InfoBlockSize;
    LONG    AvailSpace;
    LONG    UsedSpace;
    BYTE    SMString;
} SUPPORT_MODULE_INFO;
```

The `IOStatus` field contains the read and write access for the support module.

The `InfoBlockSize` field contains the size of the information block containing information about the support device. This information block follows the `SMString` field.

The `AvailSpace` field contains the amount of available space on the support module. The `UsedSpace` field contains the amount of used space on the support module.

The `SMString` contains the name of the support module and is followed by an information block. The size of `SMString` is limited to 128 bytes.

NWIsDataMigrationAllowed

Determines whether data migration is allowed for a given volume

Local Servers: nonblocking

Remote Servers: N/A

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>
```

```
LONG NWIsDataMigrationAllowed (  
    LONG    Volume);
```

Parameters

Volume

(IN) Specifies the volume number that you want information for.

Return Values

NOTE: This function does not have a cross-platform counterpart.

This function returns TRUE if data migration is allowed, or FALSE if data migration is not allowed.

NWMoveFileFromDM

Moves a file from on-line long-term storage media to a NetWare volume

Local Servers: blocking

Remote Servers: blocking

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>
```

```
LONG NWMoveFileFromDM (  
    char    *path,  
    LONG    NameSpace);
```

Parameters

path

(IN) Points to the path of the file.

NameSpace

(IN) Specifies the name space of the path.

Return Values

0	Successful.
---	-------------

Remarks

For cross-platform functionality, call [NWMoveFileFromDM \(page 32\)](#).

See Also

[NWMoveFileToDM \(page 649\)](#)

NWMoveFileToDM

Moves a file to on-line long-term data storage media while leaving the file visible on the NetWare volume

Local Servers: blocking

Remote Servers: blocking

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>

LONG NWMoveFileToDM (
    char    *path,
    LONG    NameSpace,
    LONG    SupportModuleID);
```

Parameters

path

(IN) Points to the path of the file.

NameSpace

(IN) Specifies the name space of the path.

SupportModuleID

(IN) Specifies the assigned ID number of the support module that is to migrate the data to long-term storage.

Return Values

0	Successful.
---	-------------

Remarks

This function moves a file's data to long-term storage while leaving the file visible on the NetWare volume. In this way, large, seldom-used files can be moved from the NetWare volume and put into long-term storage while not in use, yet the user can still see them on the NetWare volume.

For cross-platform functionality, call [NWMoveFileToDM \(page 34\)](#).

See Also

[NWMoveFileFromDM \(page 648\)](#)

NWPeekFileData

Enables the developer to look at data in a migrated file

Local Servers: blocking

Remote Servers: N/A

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>
```

```
LONG NWPeekFileData (  
    char    *path,  
    LONG    nameSpace,  
    LONG    dataStreamNumber,  
    LONG    startingSector,  
    LONG    sectorsToRead,  
    BYTE    *buffer,  
    LONG    *sectorsRead,  
    LONG    *bytesRead);
```

Parameters

path

(IN) Specifies the path of the file from which to read data.

nameSpace

(IN) Specifies the name space of the file (see [Section 20.5, “Name Space Flag Values,”](#) on [page 595](#)).

dataStreamNumber

(IN) Specifies the data stream for the data.

startingSector

(IN) Specifies the sector to start reading from.

sectorsToRead

(IN) Specifies the number of sectors to read.

buffer

(OUT) Points to the buffer containing the data that was read.

sectorsRead

(OUT) Receives the number of sectors read.

bytesRead

(OUT) Receives the total number of bytes read.

Return Values

0	Successful.
---	-------------

Remarks

NOTE: This function does not have a cross-platform counterpart.

This function allows the developer to read from a migrated file.

The `nameSpace` parameter can have the following values:

- 0 DOS
- 1 MACINTOSH
- 2 NFS
- 3 FTAM
- 4 LONG
- 5 NT

NWSetDefaultSupportModule

Sets the default read write support module ID

Local Servers: blocking

Remote Servers: blocking

Classification: 4.x, 5.x, 6.x

Service: Server-Based Data Migration

Syntax

```
#include <\nlm\nit\nwdatamg.h>

LONG NWSetDefaultSupportModule (
    LONG    newSupportModuleID,
    LONG    *currentSupportModuleID);
```

Parameters

newSupportModuleID

(IN) Specifies the assigned ID number of the data migration support module to migrate the data.

currentSupportModuleID

(IN) Specifies the ID number of the current support module.

Return Values

0	Successful.
---	-------------

Remarks

For cross-platform functionality, call [NWSetDefaultSupportModule \(page 37\)](#).

See Also

[NWGetDefaultSupportModule \(page 641\)](#)

Server-Based File System Functions

26

This documentation alphabetically lists the Server-Based File System functions and describes their purpose, syntax, parameters, and return values.

- ◆ “AddSpaceRestrictionForDirectory” on page 654
- ◆ “AddTrustee” on page 656
- ◆ “AddUserSpaceRestriction” on page 659
- ◆ “ChangeDirectoryEntry” on page 661
- ◆ “DeleteTrustee” on page 665
- ◆ “DeleteUserSpaceRestriction” on page 667
- ◆ “GetAvailableUserDiskSpace” on page 668
- ◆ “GetDiskSpaceUsedByObject” on page 670
- ◆ “GetEffectiveRights” on page 672
- ◆ “GetMaximumUserSpaceRestriction” on page 675
- ◆ “ModifyInheritedRightsMask” on page 677
- ◆ “PurgeTrusteeFromVolume” on page 680
- ◆ “ReturnSpaceRestrictionForDirectory” on page 681
- ◆ “ScanTrustees” on page 683
- ◆ “ScanUserSpaceRestrictions” on page 685
- ◆ “SetDirectoryInfo” on page 687
- ◆ “UpdateDirectoryEntry” on page 690

For cross-platform functionality, see [Developing NLMs with Cross-Platform Functions](#), use the CALNLM32.NLM library, and call the alternative function listed with each NLM function.

AddSpaceRestrictionForDirectory

Adds directory space restrictions

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int AddSpaceRestrictionForDirectory (
    char    *pathName,
    int     restriction,
    LONG    allowWildCardsFlag);
```

Parameters

pathName

(IN) Specifies the pathname of the directory to which to add space restrictions.

restriction

(IN) Specifies the number of 4K blocks that the files in the specified directory tree are allowed to occupy.

allowWildCardsFlag

(IN) Indicates whether or not wildcards are allowed in the pathname:

Nonzero = Wildcards allowed

0 = Wildcards are not allowed

Return Values

0	ESUCCESS
NetWare Error	UNSUCCESSFUL

Remarks

To be able to add space restrictions to a directory, you must have supervisory rights to the directory or directories being modified.

A restriction in a directory means that all the files in that directory plus all of the files in any subdirectories of that directory are not allowed to occupy more space than the amount specified by the `restriction` parameter. The space restriction value is rounded up to a multiple of 4K (4096).

Wildcard specifiers can be used to apply a disk space restriction to more than one directory at a time.

A space restriction can be removed from a directory by setting the restriction amount to zero.

`SetCurrentNameSpace` sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare versions before 4.x, this function only works with DOS name space for remote servers.

See Also

[ReturnSpaceRestrictionForDirectory \(page 681\)](#)

AddTrustee

Adds a trustee to a directory's or file's trustee list

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int AddTrustee (
    char    *pathName,
    LONG    trusteeObjectID,
    WORD    trusteeRightsMask);
```

Parameters

pathName

(IN) Specifies the string containing the path specification (maximum 255 characters, including the NULL terminator).

trusteeObjectID

(IN) Specifies the unique object ID of the trustee, in reverse order.

trusteeRightsMask

(IN) Specifies the trustee rights to assign to the directory or file.

Return Values

0x00	ESUCCESS
0x8C	ERR_NO_MODIFY_PRIVILEGES
0xFC	ERR_NO_SUCH_BINDERY_OBJECT

Remarks

This function adds a trustee to a directory's or file's trustee list by passing the trustee's object ID and an associated trustee rights mask. (Trustees can be set for files in NetWare 3.x and 4.x, unlike NetWare 2.x.) The application can obtain an object's ID and the user's object ID number by using the Directory Services function NWDSMapNameToID .

This function specifies the directory or file by passing a pathname. The `pathName` parameter can identify an absolute or relative directory or file path. An absolute path includes a volume. Examples of absolute pathnames would be:

`volume:directory\...\directory\filename`

`volume:filename`

`volume:` (equivalent to `volume:\`)

Applications can use a relative file path to specify a directory or file. The relative path, combined with the CWD specifies an absolute file path. For example, if the CWD points to `SYS:\` and the specified pathname is `PUBLIC\WORDP` or `PUBLIC\WORDP\ABC.TXT`, then in the former case, the resulting directory is `SYS:PUBLIC\WORDP` and in the latter case, `SYS:PUBLIC\WORDP\ABC.TXT`.

`AddTrustee` expects the trustee ID in reverse order (`0010000e = e0000100`) to perform properly.

The `trusteeRightsMask` parameter specifies a user's trustee rights. The bits in a trustee rights mask are defined as follows:

- 0 Read (file reads allowed)
- 1 Write (file writes allowed)
- 2 Reserved
- 3 Create (files can be created)
- 4 Delete (files can be deleted)
- 5 Access control (trustee rights can be assigned)
- 6 See files (files can be viewed in directory scan)
- 7 Modify (files can be modified)
- 8 Supervisor (all rights are granted)

The following constants have been defined for each right which can be ORed (`|`) together for a complete specification: `TA_READ`, `TA_WRITE`, `TA_CREATE`, `TA_DELETE`, `TA_ACCESSCONTROL`, `TA_SEEFILES`, `TA_MODIFY`, `TA_SUPERVISOR`.

For versions of NetWare previous to 3.0, the trustee rights appear in a 1-byte format as follows:

- 0 Read (file reads allowed)
- 1 Write (file writes allowed)
- 2 Open
- 3 Create (files can be created)
- 4 Delete (files can be deleted)
- 5 Parental (subdirectories can be created/deleted and trustee rights granted/revoked)
- 6 Search (directory can be searched)
- 7 Modify (file attributes can be modified)

Given the following path, where `component1` through `componentn-1` are directories, and `componentn` is either a file or directory:

`volume:component1\component2\...\componentn`

An object's effective rights to a file or in a directory can be determined, using the following algorithm:

- ◆ Initialize an object's effective rights to whatever rights are granted to the current connection in the root of the specified volume.
- ◆ For each component (component1 through componentn), the effective rights are intersected (ANDed) with the component's inherited rights mask.
- ◆ If the current connection is granted any rights (is a trustee) in the component, then the effective rights are ORed (|) together with the rights granted to the current connection in the component.

To be added as a trustee, a user must exist as an object. The rights mask of a new trustee is made equal to `trusteeRightsMask`. If the user is already a trustee in the specified directory or file, the existing rights mask for the trustee is replaced by the `trusteeRightsMask`.

The current connection must have access control rights to the directory or file whose trustee list is being manipulated.

`SetCurrentNameSpace` sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare versions before 4.x, this function only works with DOS name space for remote servers.

See Also

[DeleteTrustee](#) (page 665), [NWDSMapNameToID](#) (*NDK: Novell eDirectory Core Services*)

AddUserSpaceRestriction

Adds a user space restriction

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int AddUserSpaceRestriction (
    int    volume,
    LONG   trusteeID,
    LONG   restriction);
```

Parameters

volume

(IN) Specifies the volume number of the volume where the restriction is to be added (-1 specifies the current volume).

trusteeID

(IN) Specifies the trustee's object ID.

restriction

(IN) Specifies the number of 4K blocks on the disk that the user is allowed to occupy on the volume.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
152	(0x98)	ERR_INVALID_VOLUME

If `trusteeID` is invalid, no error code is returned.

Remarks

This function is used to add disk space restrictions to an object. The `restriction` parameter specifies the total disk space that an object is to have on the volume.

The value of `restriction` is a number of disk sectors. The value of `restriction` is a 4K multiplier. That is, a value of 5 indicates a disk space restriction of 20K (4K X 5 = 20K).

If user A has a disk space restriction of 500 and this function is called with a value of 1000, then user A now has a disk space restriction of 1000 not 1500.

`AddUserSpaceRestriction` is not supported in a NetWare 2.x environment. On remote servers running NetWare 2.x, this function returns error code 251 (ERR_UNKNOWN_REQUEST).

See Also

[DeleteUserSpaceRestriction \(page 667\)](#), [GetAvailableUserDiskSpace \(page 668\)](#)

ChangeDirectoryEntry

Changes a directory or file entry

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int ChangeDirectoryEntry (
    char                *pathName,
    struct ModifyStructure *modifyVector,
    LONG                modifyBits,
    LONG                allowWildCardsFlag);
```

Parameters

pathName

(IN) Specifies the directory pathname to be changed.

modifyVector

(IN) Points to a structure that specifies the new values of the directory entry's fields.

modifyBits

(IN) Tells the function which structure fields to change.

allowWildCardsFlag

(IN) Indicates whether wildcards are allowed in the pathname:

Nonzero = Wildcards allowed

0 = No wildcards allowed.

Return Values

Value	Hex	Constant and Definition
0	(0x00)	ESUCCESS
1	(0x01)	Invalid MOwnerID, MLastUpdatedID, MLastArchivedID, or MMaximumSpace in ModifyStructure.
NetWare Error		UNSUCCESSFUL

Remarks

This function is used to modify the fields of a file or directory entry or entries. (If wildcards are specified, then only matching files are changed.)

To call this function, complete the following steps:

1. Indicate which fields to change out by switching on the appropriate bit in the `modifyBits` parameter.

The modify bits are defined in `NWFATTR.H` and have the following values:

```
0x0001L MModifyNameBit
0x0002L MFileAttributesBit
0x0004L MCreateDateBit
0x0008L MCreateTimeBit
0x0010L MOwnerIDBit
0x0020L MLastArchivedDateBit
0x0040L MLastArchivedTimeBit
0x0080L MLastArchivedIDBit
0x0100L MLastUpdatedDateBit
0x0200L MLastUpdatedTimeBit
0x0400L MLastUpdatedIDBit
0x0800L MLastAccessedDateBit
0x1000L MInheritanceRestrictionMaskBit
0x2000L MMaximumSpaceBit
0x4000L MLastUpdatedInSecondsBit
```

2. Create or fill in the structure `ModifyStructure`. It is only necessary to fill in those fields to be changed (with the exception of `MFileAttributesMask`, see below). This structure is located in `NWDIR.H` and contains the following fields:

```
BYTE    *MModifyName;
LONG    MFileAttributes;
LONG    MFileAttributesMask;
WORD    MCreateDate;
WORD    MCreateTime;
LONG    MOwnerID;
WORD    MLastArchivedDate;
WORD    MLastArchivedTime;
LONG    MLastArchivedID;
WORD    MLastUpdatedDate;
WORD    MLastUpdatedTime;
LONG    MLastUpdatedID;
WORD    MLastAccessedDate;
WORD    MInheritanceGrantMask;
WORD    MInheritanceRevokeMask;
int     MMaximumSpace;
LONG    MLastUpdatedInSeconds;
```

The `MMaximumSpace` field contains the number of 4K blocks.

The `MOwnerID`, `MLastArchivedID`, and `MLastUpdatedID` must be in low-high order.

The `MFileAttributesMask` field must be set to whatever the file's current attributes are if you want to retain the existing file attributes in addition to the attributes you specify in the `MFileAttributes` field. Set the mask to -1 if you want to be able to set any file attribute.

3. Call the function.

The current connection must have the following access rights to change the specified directory entry fields:

Attribute/Field	Required Access Rights
ReadOnly	ModifyEntry
Hidden	ModifyEntry
System	ModifyEntry
ExecuteOnly	CreateNewEntry or ModifyEntry
Subdirectory	Cannot be modified
Archive	ModifyEntry
Share	
Transaction	ModifyEntry
ReadAudit	SupervisorPrivileges (over owner of file or directory)
WriteAudit	SupervisorPrivileges (over owner of file or directory)
ImmediatePurge	DeleteExistingEntry
MCreateDate	SupervisorPrivileges
MCreateTime	SupervisorPrivileges
MOwnerID	SupervisorPrivileges (over current and new owner)
MLastArchivedDate	ReadExistingFile or ModifyEntry
MLastArchivedTime	ReadExistingFile or ModifyEntry
MLastArchivedID	ReadExistingFile or ModifyEntry to set own ID; SupervisorPrivileges over current LastArchivedID to set ID of another object
MLastUpdatedDate	ModifyEntry or WriteExistingFile
MLastUpdatedTime	ModifyEntry or WriteExistingFile
MLastUpdatedID	ModifyEntry or WriteExistingFile to set own ID; SupervisorPrivileges over current LastUpdatedID to set ID of another object
MRightsGrantMask	ChangeAccessControl; cannot disinherit Supervisor Privileges
MRightsRevokeMask	ChangeAccessControl; cannot disinherit SupervisorPrivileges
MMaximumSpace	SupervisorPrivileges

`ChangeDirectoryEntry` is supported in a NetWare 2.x environment for directories only, and can only change attributes, create date and time, inherited rights, and owner ID. File entries under NetWare 2.x must still be set using `SetFileInfo` .

SetCurrentNameSpace sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare versions before 4.x, this function only works with DOS name space for remote servers.

See Also

[ModifyInheritedRightsMask \(page 677\)](#), [SetDirectoryInfo \(page 687\)](#), [SetFileInfo](#)

DeleteTrustee

Removes a trustee from a directory's or file's trustee list

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int DeleteTrustee (
    char    *pathName,
    LONG    trusteeObjectID);
```

Parameters

pathName

(IN) Specifies the string containing path specification (maximum 255 characters, including the NULL terminator).

trusteeObjectID

(IN) Specifies the unique object ID of trustee.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
152	(0x98)	ERR_VOLUME_DOES_NOT_EXIST
156	(0x9C)	ERR_INVALID_PATH

Remarks

The DeleteTrustee function revokes all of the rights that a trustee has been granted. This function specifies the trustee by passing the trustee's object ID. The function identifies the directory or file by optionally passing a complete pathname or a partial pathname relative to the current working directory (CWD). In order to delete a trustee, the current connection must have access control rights to the directory or file.

This function specifies the directory or file by passing a pathname. The pathName parameter can identify an absolute or relative directory or file path.

An absolute path includes a volume. Examples of absolute pathnames would be:

volume:directory\...\directory\filename

volume:filename

volume: (equivalent to volume:\)

Applications might use a relative file path to specify a directory or file. The relative path, combined with the CWD specifies an absolute file path. For example, if the CWD points to SYS:\ and the specified pathname is PUBLIC\WORDP or PUBLIC\WORDP\ABC.TXT, then in the former case, the resulting directory is SYS:PUBLIC\WORDP and in the latter case, SYS:PUBLIC\WORDP\ABC.TXT.

The application can obtain an object's ID by using `NWDSMapNameToID` or `ScanTrustees` .

`SetCurrentNameSpace` sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare versions before 4.x, this function only works with DOS name space for remote servers.

See Also

[AddTrustee \(page 656\)](#), [NWDSMapNameToID](#), [ScanTrustees \(page 683\)](#)

DeleteUserSpaceRestriction

Deletes a space restriction for an object

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

SMP Aware: No

Service: Volume

Syntax

```
#include <nwdir.h>

int DeleteUserSpaceRestriction (
    int     volume,
    LONG    objectID);
```

Parameters

volume

(IN) Specifies the volume number on the volume where the user restriction is to be removed.

objectID

(IN) Specifies the user's object ID.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

This function removes a space restriction on an object.

DeleteUserSpaceRestriction is not supported in a NetWare 2.x environment. On remote servers running NetWare 2.x, this function returns error code 251 (ERR_UNKNOWN_REQUEST).

See Also

[AddUserSpaceRestriction](#) (page 659), [AddSpaceRestrictionForDirectory](#) (page 654), [GetVolumeNumber](#) (Volume Management), [ReturnSpaceRestrictionForDirectory](#) (page 681)

GetAvailableUserDiskSpace

Returns the disk space available to a user in blocks

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

SMP Aware: No

Service: Volume

Syntax

```
#include <nwdir.h>

int GetAvailableUserDiskSpace (
    char    *pathName,
    LONG    *availableSpace);
```

Parameters

pathName

(IN) Points to the directory pathname that the available disk space is to be returned for.

availableSpace

(OUT) Points to the remaining disk space, in blocks, available to the user in the specified directory.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

This function returns the amount of disk space (in blocks) in the specified directory for the current connection. The disk space returned also includes purgeable blocks. The amount of space available is limited in three ways:

- ◆ User space restriction (the "user" is specified by the current connection)
- ◆ Directory space restriction
- ◆ Physical space left on the volume

GetAvailableUserDiskSpace is not supported in a NetWare 2.x environment. On remote servers running NetWare 2.x, this function returns error code 251 (ERR_UNKNOWN_REQUEST).

SetCurrentNameSpace sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare versions before 4.x, this function only works with DOS name space for remote servers.

See Also

[DeleteUserSpaceRestriction \(page 667\)](#), [ReturnSpaceRestrictionForDirectory \(page 681\)](#)

GetDiskSpaceUsedByObject

Returns the disk space being used by a particular user

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

SMP Aware: No

Service: Volume

Syntax

```
#include <nwdir.h>

int GetDiskSpaceUsedByObject (
    long    trusteeID,
    int     volume,
    LONG    *usedSpace);
```

Parameters

trusteeID

(IN) Specifies the desired user object ID.

volume

(IN) Specifies the desired volume.

usedSpace

(OUT) Receives the number of 4K blocks being used by the user object.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
Nonzero		Invalid volume or user object ID.

See Also

[AddTrustee \(page 656\)](#), [DeleteTrustee \(page 665\)](#), [ModifyInheritedRightsMask \(page 677\)](#)

Example

```
#include <stdlib.h>
#include <nwdir.h>

main()
{
    int    rc;
    long   objectID;
    LONG   usedSpace;

    rc = GetBinderyObjectID("dgambill", 1, &objectID);
    if( rc != 0)
    {
        printf("GetBinderyObjectID() status = %x\n", rc);
        return;
    }
    rc = GetDiskSpaceUsedByObject( objectID, 0, &usedSpace);
    if( rc != 0)
    {
        printf("GetDiskSpaceUsedByObject() status = %x\n", rc);
        return;
    }
    printf("Disk Space Used By 'dgambill' = %d\n",
           usedSpace*4096);
}
```

GetEffectiveRights

Returns the current connection's effective rights to a directory or file

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int GetEffectiveRights (
    char    *pathName,
    WORD    *effectiveRightsMask);
```

Parameters

pathName

(IN) Specifies the string containing the path specification (maximum 255 characters, including the NULL terminator).

effectiveRightsMask

(OUT) Returns the current connection's rights to the specified directory or file.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
152	(0x98)	ERR_VOLUME_DOES_NOT_EXIST
191	(0xBF)	ERR_INVALID_NAMESPACE

Remarks

This function specifies the directory or file by passing a pathname. The `pathName` parameter can identify an absolute or relative directory or file path. An absolute path includes a volume. Examples of absolute pathnames would be:

volume:directory\...\directory\filename

volume:filename

volume: (equivalent to volume:\)

Applications can use a relative file path to specify a directory or file. The relative path, combined with the CWD specifies an absolute file path. For example, if the CWD points to SYS:\ and the specified pathname is PUBLIC\WORDP or PUBLIC\WORDP\ABC.TXT, then in the former case, the resulting directory is SYS:PUBLIC\WORDP and in the latter case, SYS:PUBLIC\WORDP\ABC.TXT.

The `effectiveRightsMask` parameter returns a user's effective rights to the specified directory or file.

Given the following path, where `component1` through `componentn-1` are directories, and `componentn` is either a file or directory:

```
volume:component1\component2\...\componentn
```

A user's effective rights to a file or in a directory can be determined using the following algorithm.

- ◆ Initialize the user's effective rights to whatever rights are granted to the current connection in the root of the specified volume.
- ◆ For each component (`component1` through `componentn`), the effective rights are intersected (ANDed) with the component's inherited rights mask.
- ◆ If the current connection is granted any rights (is a trustee) in the component, then the effective rights are ORed (|) together with the rights granted to the current connection in the component.

For NetWare 3.x and 4.x, the bits in an effective rights mask are defined as follows:

- 0 Read (file reads allowed)
- 1 Write (file writes allowed)
- 2 Reserved
- 3 Create (files can be created)
- 4 Delete (files can be deleted)
- 5 Access control (trustee rights can be assigned)
- 6 See files (files can be viewed in directory scan)
- 7 Modify (files can be modified)
- 8 Supervisor (all rights are granted)

For versions of NetWare previous to 3.0, the trustee rights appear in a 1-byte format as follows:

- 0 Read (file reads allowed)
- 1 Write (file writes allowed)
- 2 Open
- 3 Create (files can be created)
- 4 Delete (files can be deleted)
- 5 Parental (subdirectories can be created/deleted and trustee rights granted/revoked)
- 6 Search (directory can be searched)
- 7 Modify (file attributes can be modified)

`SetCurrentNameSpace` sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare versions before 4.x, this function only works with DOS name space for remote servers.

See Also

[AddTrustee \(page 656\)](#), [DeleteTrustee \(page 665\)](#), [ModifyInheritedRightsMask \(page 677\)](#)

GetMaximumUserSpaceRestriction

Returns the maximum disk space restriction for a particular user

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM

SMP Aware: No

Service: Volume

Syntax

```
#include <nwdir.h>

int GetMaximumUserSpaceRestriction (
    long trusteeID,
    int volume,
    LONG *maxRestriction);
```

Parameters

trusteeID

(IN) Specifies the desired user object ID.

volume

(IN) Specifies the desired volume (0-63 for NetWare 3.1 and later; 0-31 for previous versions).

maxRestriction

(OUT) Receives the number of 4K blocks to which the user is restricted. If this value is 0, there is no restriction.

Return Values

Value	Hex	Constant
400000000H	(0x00)	ESUCCESS
Nonzero		Invalid volume, user object ID, or network error.

Remarks

GetMaximumUserSpaceRestriction is *not* supported in a NetWare 2.x environment. Remote servers running NetWare 2.x return error code 251 (ERR_UNKNOWN_REQUEST).

See Also

[AddTrustee \(page 656\)](#), [DeleteTrustee \(page 665\)](#), [ModifyInheritedRightsMask \(page 677\)](#)

Example

```
#include <stdlib.h>
#include <nwdir.h>

main()
{
    int    rc;
    long   objectID;
    LONG   maxRestriction;
    rc = GetBinderyObjectID("testuser", 1, &objectID);
    if( rc != 0)
    {
        printf("GetBinderyObjectID() status = %x\n", rc);
        return;
    }
    rc = GetMaximumUserSpaceRestriction( objectID, 0,
        &maxRestriction);
    if( rc != 0)
    {
        printf("GetMaximumUserSpaceRestriction() status = %x\n", rc);
        return;
    }
    printf("Max Disk Space Restriction for 'testuser' = %d\n",
        maxRestriction*4096);
}
```

ModifyInheritedRightsMask

Modifies the inherited rights mask of a directory or file

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int ModifyInheritedRightsMask (
    char *path,
    WORD  revokeRightsMask,
    WORD  grantRightsMask);
```

Parameters

path

(IN) Specifies the string containing the path specification for the directory or file to be modified (maximum 255 characters, including the NULL terminator).

revokeRightsMask

(IN) Specifies the rights mask that specifies which rights in the directory's inherited rights mask are to be modified.

grantRightsMask

(IN) Specifies the rights mask to receive the modified rights.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
140	(0x8C)	ERR_NO_MODIFY_PRIVILEGES
152	(0x98)	ERR_VOLUME_DOES_NOT_EXIST
156	(0x9C)	ERR_INVALID_PATH

Remarks

For remote server support, this function returns the maximum rights mask for NetWare 2.x.

The `ModifyInheritedRightsMask` function specifies the directory or file by passing a pathname. The `path` parameter can identify an absolute or relative directory or file path. An absolute path includes a volume.

Examples of absolute pathnames would be:

`volume:directory\...\directory\filename`

`volume:filename`

`volume:` (equivalent to `volume:\`)

Applications can use a relative file path to specify a directory or file. The relative path, combined with the CWD, specifies an absolute file path. For example, if the CWD points to `SYS:\` and the specified pathname is `PUBLIC\WORDP` or `PUBLIC\WORDP\ABC.TXT`, then in the former case, the resulting directory is `SYS:PUBLIC\WORDP` and in the latter case, `SYS:PUBLIC\WORDP\ABC.TXT`.

The function specifies which rights to modify by passing the `revokeRightsMask`.

Both the `grantRightsMask` and the `revokeRightsMask` are 1-WORD parameters with bits defined as follows:

- 0 Read (file reads allowed)
- 1 Write (file writes allowed)
- 2 Reserved
- 3 Create (files can be created)
- 4 Delete (files can be deleted)
- 5 Access control (trustee rights can be assigned)
- 6 See files (files can be viewed in directory scan)
- 7 Modify (files can be modified)
- 8 Supervisor (all rights are granted)

The `grantRightsMask` and the `revokeRightsMask` parameters are both single-byte parameters for NetWare 2.x remote server support. There is no Supervisor bit for NetWare 2.x servers.

- 0 Read (file reads allowed)
- 1 Write (file writes allowed)
- 2 Open
- 3 Create (files can be created)
- 4 Delete (files can be deleted)
- 5 Parental (subdirectories can be created/deleted and trustee rights granted/revoked)
- 6 Search (directory can be searched)
- 7 Modify (file attributes can be modified)

The rights in the directory's inherited rights mask are modified according to the `revokeRightsMask` and are placed in the `grantRightsMask`. The inherited rights mask can be completely reset by setting the `revokeRightsMask` to `0xFF` and then setting the `grantRightsMask` to the desired inherited rights mask.

The current connection must have access control rights to the directory or file whose inherited rights mask is being modified.

SetCurrentNameSpace sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare versions before 4.x, this function only works with DOS name space for remote servers.

See Also

[GetEffectiveRights \(page 672\)](#)

PurgeTrusteeFromVolume

Deletes a trustee from a volume

Local Servers: blocking

Remote Servers: N/A

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int PurgeTrusteeFromVolume (
    int    volume,
    LONG   trusteeID);
```

Parameters

volume

(IN) Specifies the volume number of the volume from which to remove all trustee references.

trusteeID

(IN) Specifies the trustee's object ID.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

The PurgeTrusteeFromVolume function deletes all references to trustee from a volume. It does not perform a security check based on the current connection. After this function call is made, the trustee no longer has any rights on the specified volume.

See Also

[DeleteTrustee \(page 665\)](#)

ReturnSpaceRestrictionForDirectory

Returns space restrictions for a directory

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int ReturnSpaceRestrictionForDirectory (
    char    *pathName,
    LONG    numberOfStructuresToReturn,
    BYTE    *answerBuffer,
    LONG    *numberOfStructuresReturned);
```

Parameters

pathName

(IN) Specifies the pathname of directory for which to get space restrictions.

numberOfStructuresToReturn

(IN) Specifies the number of answer structures (9-byte) that `answerBuffer` can hold.

answerBuffer

(OUT) Receives the space restriction information for the directory.

numberOfStructuresReturned

(OUT) Receives the actual number of structures placed in `answerBuffer`.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

This function returns space restrictions for a directory and all of its parent directories.

The result placed in `answerBuffer` is an array of structures. Each structure has the following format (defined in `NWDIR.H`):

Offset	Content	Type
0	<code>ALevelNumber</code>	BYTE
1	<code>AMaximumAmount</code>	LONG
5	<code>ACurrentAmount</code>	LONG

The `ALevelNumber` field specifies the depth into the directory. For example, the level number for the directory `SYS:ONE\TWO\THREE` is 3.

The `AMaximumAmount` field specifies the space restriction for a directory.

The `ACurrentAmount` field specifies the amount of space available at the time of the call. This field receives the number of 4K restrictions.

If there is no space restriction, `AMaximumAmount` is `0x7FFFFFFF`.

The `numberOfStructuresToReturn` parameter should be at least as large as the number of levels that the directory is deep in the directory structure. The reason for this is that this function returns the space restriction for all of the parent directories as well.

`ReturnSpaceRestrictionForDirectory` is not supported in a NetWare 2.x environment. Remote servers running NetWare 2.x return error code 251 (`ERR_UNKNOWN_REQUEST`).

`SetCurrentNameSpace` sets the name space which is used for parsing the path input to this function.

See Also

[AddSpaceRestrictionForDirectory \(page 654\)](#)

ScanTrustees

Returns information about directory or file trustees

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int ScanTrustees (
    char    *pathName,
    LONG    startingOffset,
    LONG    vectorSize,
    LONG    *trusteeVector,
    WORD    *maskVector,
    LONG    *actualVectorSize);
```

Parameters

pathName

(IN) Specifies the string containing the path specification for the directory to be scanned (maximum 255 characters, included the NULL terminator).

startingOffset

(IN) Specifies the starting byte.

vectorSize

(IN) Specifies the Number of trusteeVector structures that trusteeVector can hold.

trusteeVector

(OUT) Points to an array of structures containing the trustees of the scanned directory.

maskVector

(OUT) Points to structure that specifies the trustee rights.

actualVectorSize

(OUT) Receives the actual number of trusteeVector structures being returned.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

An application can use this function iteratively to scan a directory and return information about all the directory trustees.

SetCurrentNameSpace sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare versions before 4.x, this function only works with DOS name space for remote servers.

It's an ID that can be converted into NWDSMapIDToName()

Example:

```
ScanTrustees(path, startingOffset, TRUSTEES_PER_SCAN, trusteeVector,  
maskVector, &actualVectorSize)
```

```
for (i = 0; i < actualVectorSize; i++)  
{  
    char name[MAX_DN_CHARS + 1];  
    LONG trustee = NWLongSwap(trusteeVector[i]);  
    ccode = NWDSMapIDToName(context, conn, trustee, name);  
}
```

See Also

ScanBinderyObjectTrusteePaths

ScanUserSpaceRestrictions

Returns information about users' space restrictions on a volume

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>
```

```
int ScanUserSpaceRestrictions (  
    int     volumeNumber,  
    LONG    *sequenceNumber,  
    LONG    numberOfTrusteesToReturn,  
    LONG    *answerArea,  
    LONG    *numberOfTrusteesReturned);
```

Parameters

volumeNumber

(IN) Specifies the volume number of the volume to be searched (0-63 for NetWare 3.1 and later; 0-31 for previous versions).

sequenceNumber

(IN/OUT) The initial search requires a 0 as input; after the initial search, the sequence number is incremented automatically within the function so the user only needs to initialize once.

numberOfTrusteesToReturn

(IN) Specifies the number of trustees to scan for.

answerArea

(OUT) Points to the buffer in which to place the result (returned by `numberOfTrusteesReturned`).

numberOfTrusteesReturned

(OUT) Returns the number of trustees for which space restriction information has been retrieved.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
152	(0x98)	ERR_VOLUME_DOES_NOT_EXIST

Remarks

An application can use this function to return information about space restrictions for trustees. The function scans for as many trustees as specified by the `numberOfTrusteesToReturn` parameter.

The `answerArea` parameter points to an array of structures. Each structure has the following format:

```
LONG trusteeID;  
LONG restriction;
```

The `restriction` field contains the space restriction in 4K blocks.

The CWV is used if the input `volumeNumber` is set to -1.

See Also

[AddUserSpaceRestriction \(page 659\)](#), [DeleteUserSpaceRestriction \(page 667\)](#)

SetDirectoryInfo

Changes a directory's information

Local Servers: blocking

Remote Servers: blocking

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int SetDirectoryInfo (
    char *directoryPath,
    BYTE *newCreationDateAndTime,
    LONG newOwnerObjectID,
    WORD newInheritedRightsMask);
```

Parameters

directoryPath

(IN) Specifies the string containing the path for the directory whose information is changed (maximum 255 characters, including the NULL terminator).

newCreationDateAndTime

(IN) Specifies the date and time that the directory was created (standard DOS format, 4 bytes).

newOwnerObjectID

(IN) Specifies the unique object ID of the new owner of the directory.

newInheritedRightsMask

(IN) Specifies the new inherited rights mask of the directory.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
191	(0xBF)	ERR_INVALID_NAME_SPACE
NetWare Error		UNSUCCESSFUL

Remarks

The `newInheritedRightsMask` parameter only specifies additional rights to be granted. Call `ModifyInheritedRightsMask` to revoke rights.

This function specifies a creation date and time, owner object ID, and inherited rights mask. The function defines the target directory by passing a partial or complete path.

volume:directory\...\directory\filename

volume:filename

volume: (equivalent to volume:)

Applications can use a relative file path to specify a directory or file. The relative path, combined with the CWD, specifies an absolute file path. For example, if the CWD points to `SYS:\` and the specified pathname is `PUBLIC\WORDP` or `PUBLIC\WORDP\ABC.TXT`, then in the former case, the resulting directory is `SYS:PUBLIC\WORDP` and in the latter case, `SYS:PUBLIC\WORDP\ABC.TXT`.

The `creationDateAndTime` parameter appears in standard DOS format as follows:

Figure 26-1 Date and Time Fields

Byte 1								Byte 0							
Year								Month				Day			
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 3								Byte 2							
Hour				Minute				Seconds x 2							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

The function returns the date and time in ascending order (byte 1, byte 2, byte 3, byte 4).

The `newOwnerObjectID` parameter contains the object ID of the directory owner.

The `newInheritedRightsMask` parameter contains the directory's inherited rights mask. The bits in the inherited rights mask are defined as follows:

- 0 Read (file reads allowed)
- 1 Write (file writes allowed)
- 2 Reserved
- 3 Create (files can be created)
- 4 Delete (files can be deleted)
- 5 Access control (trustee rights can be assigned)
- 6 See files (files can be viewed in directory scan)
- 7 Modify (files can be modified)
- 8 Supervisor (all rights are granted)

NOTE: The `newInheritedRightsMask` parameter for NetWare 2.x remote server support is actually the `newMaximumRightsMask`. The parameter is a single-byte value and there is no Supervisor bit in the Maximum Rights Mask for a NetWare 2.x server.

The following constants have been defined for each right which can be ORed (|) together for a complete specification:

TA_READ
TA_WRITE
TA_CREATE
TA_DELETE
TA_ACCESSCONTROL
TA_SEEFILES
TA_MODIFY
TA_SUPERVISOR

To change a directory's information, the current connection must have access control and modify rights to the directory's parent.

The SUPERVISOR or supervisor equivalent are the only users that can change the owner of a directory.

`SetCurrentNameSpace` sets the name space which is used for parsing the path input to this function.

NOTE: For NetWare versions before 4.x, this function only works with DOS name space for remote servers.

UpdateDirectoryEntry

Updates a directory entry

Local Servers: blocking

Remote Servers: N/A

Classification: 3.x, 4.x, 5.x, 6.x

SMP Aware: No

Service: File System

Syntax

```
#include <nwdir.h>

int UpdateDirectoryEntry (
    int handle);
```

Parameters

handle

(IN) Specifies a file handle obtained from an open or creat.

Return Values

Value	Hex	Constant
0	(0x00)	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

This function updates the target file's file entry in the directory table with the current file size, current date and time, and File Allocation Table (FAT) chain information. The updated information is not actually written to disk until sometime after the function returns.

Revision History

A

The following table outlines all the changes that have been made to the Multiple and Inter-File Management documentation (in reverse chronological order):

February 28, 2007	Updated NWOpenNSEntry (page 516) .
October 11, 2006	Updated NWIntScanForTrustees (page 244) , ScanTrustees (page 683) , and NWScanConnectionsUsingFile (page 260) .
March 1, 2006	Updated format.
October 5, 2005	Transitioned to revised Novell documentation standards.
March 2, 2005	Modified the documentation for Section 16.2, "Default Name Space," on page 428 , GetExtendedFileAttributes (page 147) , NWAddTrusteeToNSDirectory (page 448) , and NWDeleteTrusteeFromNSDirectory (page 459) .
June 9, 2004	Added some sample code for NWIntScanForTrustees (page 244) . Added the new file system, deleted file, and name space functions that are designed to use UTF-8 strings: NWAddTrusteeExt (page 156) , NWDeleteTrusteeExt (page 179) , NWGetEffectiveRightsExt (page 203) , NWAllocTempNSDirHandle2Ext (page 453) , NWGetDirectoryBaseExt (page 464) , NWScanForDeletedFilesExt (page 57) , etc.
February 18, 2004	Added Section 16.2, "Default Name Space," on page 428 . Modified the documentation for the following functions and structures: NWGetDirSpaceLimitList2 (page 197) , NWIntMoveDirEntry (page 226) , and OpenFileCallBackStruct (page 418) .
October 8, 2003	Removed the Delphi syntax for NWScanNSDirectoryForTrustees (page 530) . Delphi does not expose this function.
July 30, 2003	Fixed the Delphi syntax for SEARCH_DIR_INFO (page 363) and TRUSTEE_INFO (page 371) .
June 2003	Modified the description of FEQuickFileLength (page 99) . Added a note about NSS volumes to the GetNameSpaceName (page 442) function. Fixed the prototype for the NWScanNSEntryInfo2 (page 538) function. Fixed a typo in the ChangeDirectoryEntry (page 661) function. Changed all Pascal references to Delphi references.
March 2003	Modified the _splitpath (page 318) function to indicate that it only works with the DOS namespace.
October 2002	Modified the Pascal syntax for the structures. Modified the documentation for UnAugmentAsterisk (page 324) and FEQuickFileLength (page 99) .
September 2002	Updated the documentation for the following functions: NWScanNSEntryInfo (page 533) and NWIntMoveDirEntry (page 226)
May 2002	Updated the introduction of Chapter 16, "Name Space Concepts," on page 427 . Updated the description of iterHandle in NWSetDirEntryInfo (page 277) . Added a Pascal syntax to NWGetDirSpaceLimitList2 (page 197) .

February 2002	<p>Updated the description of <code>augmentFlag</code> in NWIntScanFileInfo2 (page 238).</p> <p>Updated the Pascal syntax of NWScanNSEntryInfo2 (page 538) and NW_ENTRY_INFO2 (page 578).</p> <p>Updated links.</p>
October 2001	<p>Added <code>fileHandle</code> to OpenFileCallbackStruct (page 418) and provided an explanation of valid fields for the <code>_PRE_</code> and <code>_POST_</code> hooks.</p> <p>Updated Pascal syntax of NW_LIMIT_LIST (page 349) and NW_NS_OPENCREATE (page 588). Added Pascal syntax for NWScanNSEntryInfo2 (page 538) and NW_ENTRY_INFO2 (page 578).</p>
September 2001	<p>Added support for NetWare 6.x to documentation.</p> <p>Added descriptions to graphics.</p>
June 2001	<p>Added table headings.</p>
February 2001	<p>Added documentation for FEQuickFileLength (page 99) and FEQuickWrite (page 105).</p> <p>Moved the following volume functions from Chapter 26, "Server-Based File System Functions," on page 653 to Volume Management:</p> <ul style="list-style-type: none"> GetNumberOfVolumes GetVolumeInformation GetVolumeInfoWithNumber GetVolumeName GetVolumeNumber GetVolumeStatistics <p>Changed getcwd (page 146) to state the the allocated string must be freed.</p> <p>Updated NWOpenDataStream (page 512) to clarify that a DOS namespace directory handle must be passed to <code>dirHandle</code> and how this parameter and <code>datastream</code> work together.</p> <p>Changed "bindery object" to "object" references since these references can also specify NDS objects.</p>
September 2000	<p>Added cross-references to NWSetNSEntryDOSInfo (page 551) to NWSetExtendedFileAttributes2 (page 283) and NWSetFileAttributes (page 286).</p>

July 2000	<p>Added UnAugmentAsterisk (page 324) and added information about that function to opendir (page 296) and readdir (page 300).</p> <p>Added values for flags parameter in FEGetOpenFileInfo (page 79) and FEGetOpenFileInfoForNS (page 82).</p> <p>Corrected several values in Chapter 20, "Name Space Values," on page 593.</p> <p>Corrected rename (page 304) to reflect that it works for the LONG name space as well as the DOS name space.</p> <p>Corrected header for UseAccurateCaseForPaths (page 326).</p> <p>Removed the following obsolete functions from the documentation:</p> <ul style="list-style-type: none"> ◆ NWPurgeErasedFiles, NWRestoreErasedFile from Chapter 5, "Deleted File Functions," on page 45 ◆ _NWConvertHandle, NWFileSearchInitialize, NWRestoreDirectoryHandle, NWSaveDirectoryHandle from Chapter 10, "File System Functions," on page 137 ◆ NWAllocTempNSDirHandle from Chapter 18, "Name Space Functions," on page 439 ◆ NWGetPathFromDirectoryEntry from Chapter 23, "Path and Drive Functions," on page 605
May 2000	<p>Added information about calling NWGetExtendedVolumeInfo to return the block size to NWGetDirSpaceInfo (page 193).</p> <p>Added 4K block information to MODIFY_DOS_INFO (page 566).</p> <p>Changed header file for _makepath (page 149) and _splitpath (page 318) to the nwfileio.h file.</p>
March 2000	<p>Changed ownerID reference in Remarks section to be <code>objectID</code> in TRUSTEE_INFO (page 371).</p> <p>Added explanation of how to reverse FERRegisterNSPathParser (page 107).</p>

January 2000	<p>Added NWScanNSEntryInfo2 (page 538) and four corresponding structures.</p> <p>Added UseAccurateCaseForPaths (page 326).</p> <p>Added NWDeleteTrusteeFromNSDirectory (page 459).</p> <p>Added sample code to NWGetCompressedFileLengths (page 186).</p> <p>Updated Remarks section of NWGetNSPath (page 489) because this function returns only the directory path even if a file name is passed.</p> <p>Updated Remarks section of NWOpenNSEntry (page 516) because NULL should be passed to <code>fileHandle</code> if a directory is being created.</p> <p>Updated Remarks section of NWModifyMaximumRightsMask (page 251) because the current rights mask value can be returned by calling <code>NWIntScanDirectoryInformation2</code>.</p> <p>Updated Remarks section of NWScanForDeletedFiles (page 54) because the function returns -2 if <code>entryInfo</code> and <code>itemHandle</code> are NULL or <code>dirHandle</code> is zero.</p> <p>Changed the description of NWVolumeIsCDROM (page 294).</p> <p>Changed the last two parameters of NWRecoverDeletedFile (page 49) to be OUT (rather than IN) parameters.</p>
November 1999	<p>Added NWScanNSEntryInfoSet (page 541).</p> <p>Added functions in “Server-Based File System Functions” on page 653 to Master API List.</p> <p>Added description for <code>newPathString</code> in RenameMoveEntryCallBackStruct (page 422).</p> <p>Added descriptions for <code>dataForkFirstFAT</code> and <code>otherForkSize</code> in NW_EXT_FILE_INFO (page 341).</p> <p>Added descriptions for <code>maximumSpace</code> in MODIFY_DOS_INFO (page 566) and <code>hugeStateInfo</code> and <code>hugeDataLength</code> in NW_NS_INFO (page 585).</p> <p>Added library information for each function.</p> <p>Updated Remarks section of NWParsePath (page 624).</p> <p>Updated Remarks section of SetFileInfo (page 313) explaining that the date/time field must be in DOS format.</p> <p>Split the Return Mask Values into two topics: Section 20.6, “Basic Return Mask Values,” on page 595 and Section 20.7, “Extended Return Mask Values,” on page 596.</p>
September 1999	<p>Added NWGetDirSpaceLimitList2 (page 197).</p> <p>Added an example of a length-preceded string that is returned in <code>pathName</code> to NWGetPathFromDirectoryBase (page 620).</p> <p>Deleted the pointer indicator from <code>ccode</code> in NWAddFSMonitorHook (page 386).</p> <p>Updated Remarks sections of NWGetDirSpaceInfo (page 193) and NWGetDirSpaceLimitList (page 195) and their related structures.</p> <p>Replaced 0x16 EBADHNDL return value with 0x04 EBADF in closedir (page 143).</p>

July 1999	Obsoleted NWSetInitDrive. Removed NWGetSearchDriveVector and NWSetSearchDriveVector (supported for DOS and Windows 3.1 only). Removed NWParseConfig and NWSetNetWareErrorMode (supported for OS/2, DOS, and Windows 3.1 only). Removed GrammarTableStruct, SetTableStruct, TypeDefaultStruct, and PARAMETER_TABLE_TYPE (used in NWParseConfig).
June 1999	Added NWGetVolumeFlags (page 216) and NWSetVolumeFlags (page 292) . Added NWAddTrusteeToNSDirectory (page 448) .
