Novell Developer Kit

www.novell.com

VOLUME MANAGEMENT

October 5, 2005

Novell®

Legal Notices

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

AppNotes is a registered trademark of Novell, Inc.
AppTester is a registered trademark of Novell, Inc., in the United States.
ASM is a trademark of Novell, Inc.
BorderManager is a registered trademark of Novell, Inc.
BrainShare is a registered service mark of Novell, Inc., in the United States and other countries.
C3PO is a trademark of Novell, Inc.
Certified Novell Engineer is a service mark of Novell, Inc.
Client32 is a trademark of Novell, Inc.
CNE is a registered service mark of Novell, Inc.
ConsoleOne is a registered trademark of Novell, Inc.
Controlled Access Printer is a trademark of Novell, Inc.
Custom 3rd-Party Object is a trademark of Novell, Inc.
DeveloperNet is a registered trademark of Novell, Inc., in the United States and other countries.
DirXML is a registered trademark of Novell, Inc.
eDirectory is a trademark of Novell, Inc.
Excelerator is a trademark of Novell, Inc.
exteNd is a trademark of Novell, Inc.
exteNd Director is a trademark of Novell, Inc.
exteNd Workbench is a trademark of Novell, Inc.
FAN-OUT FAILOVER is a trademark of Novell, Inc.
GroupWise is a registered trademark of Novell, Inc., in the United States and other countries.
Hardware Specific Module is a trademark of Novell, Inc.
Hot Fix is a trademark of Novell, Inc.
iChain is a registered trademark of Novell, Inc.
Internetwork Packet Exchange is a trademark of Novell, Inc.
IPX is a trademark of Novell, Inc.
IPX/SPX is a trademark of Novell, Inc.
jBroker is a trademark of Novell, Inc.
Link Support Layer is a trademark of Novell, Inc.
LSL is a trademark of Novell, Inc.
ManageWise is a registered trademark of Novell, Inc., in the United States and other countries.
Mirrored Server Link is a trademark of Novell, Inc.
Mono is a registered trademark of Novell, Inc.
MSL is a trademark of Novell, Inc.
My World is a registered trademark of Novell, Inc., in the United States.
NCP is a trademark of Novell, Inc.
NDPS is a registered trademark of Novell, Inc.
NDS is a registered trademark of Novell, Inc., in the United States and other countries.
NDS Manager is a trademark of Novell, Inc.
NE2000 is a trademark of Novell, Inc.
NetMail is a registered trademark of Novell, Inc.
NetWare is a registered trademark of Novell, Inc., in the United States and other countries.
NetWare/IP is a trademark of Novell, Inc.
NetWare Core Protocol is a trademark of Novell, Inc.
NetWare Loadable Module is a trademark of Novell, Inc.

NetWare Management Portal is a trademark of Novell, Inc.

NetWare Name Service is a trademark of Novell, Inc.

NetWare Peripheral Architecture is a trademark of Novell, Inc.

NetWare Requester is a trademark of Novell, Inc.

NetWare SFT and NetWare SFT III are trademarks of Novell, Inc.

NetWare SQL is a trademark of Novell, Inc.

NetWire is a registered service mark of Novell, Inc., in the United States and other countries.

NLM is a trademark of Novell, Inc.

NMAS is a trademark of Novell, Inc.

NMS is a trademark of Novell, Inc.

Novell is a registered trademark of Novell, Inc., in the United States and other countries.

Novell Application Launcher is a trademark of Novell, Inc.

Novell Authorized Service Center is a service mark of Novell, Inc.

Novell Certificate Server is a trademark of Novell, Inc.

Novell Client is a trademark of Novell, Inc.

Novell Cluster Services is a trademark of Novell, Inc.

Novell Directory Services is a registered trademark of Novell, Inc.

Novell Distributed Print Services is a trademark of Novell, Inc.

Novell iFolder is a registered trademark of Novell, Inc.

Novell Labs is a trademark of Novell, Inc.

Novell SecretStore is a registered trademark of Novell, Inc.

Novell Security Attributes is a trademark of Novell, Inc.

Novell Storage Services is a trademark of Novell, Inc.

Novell, Yes, Tested & Approved logo is a trademark of Novell, Inc.

Nsure is a registered trademark of Novell, Inc.

Nterprise is a trademark of Novell, Inc.

Nterprise Branch Office is a trademark of Novell, Inc.

ODI is a trademark of Novell, Inc.

Open Data-Link Interface is a trademark of Novell, Inc.

Packet Burst is a trademark of Novell, Inc.

PartnerNet is a registered service mark of Novell, Inc., in the United States and other countries.

Printer Agent is a trademark of Novell, Inc.

QuickFinder is a trademark of Novell, Inc.

Red Box is a trademark of Novell, Inc.

Red Carpet is a registered trademark of Novell, Inc., in the United States and other countries.

Sequenced Packet Exchange is a trademark of Novell, Inc.

SFT and SFT III are trademarks of Novell, Inc.

SPX is a trademark of Novell, Inc.

Storage Management Services is a trademark of Novell, Inc.

SUSE is a registered trademark of SUSE AG, a Novell business.

System V is a trademark of Novell, Inc.

Topology Specific Module is a trademark of Novell, Inc.

Transaction Tracking System is a trademark of Novell, Inc.

TSM is a trademark of Novell, Inc.

TTS is a trademark of Novell, Inc.

Universal Component System is a registered trademark of Novell, Inc.

Virtual Loadable Module is a trademark of Novell, Inc.

VLM is a trademark of Novell, Inc.

Yes Certified is a trademark of Novell, Inc.

ZENworks is a registered trademark of Novell, Inc., in the United States and other countries.

Third-Party Materials

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

Contents

About This Guide

The volume management functions enable you to manage and obtain statistics about NetWare® volumes. They allow you to perform the following tasks:

- Return information about a specified volume
- Access space restrictions for a specified object on a specified volume
- Access utilization statistics for a specified volume

This guide contains the following sections:

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).

Additional Information

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

- NLM Development Concepts, Tools, and Functions (http://developer.novell.com/ndk/doc/clib/ndev_enu/data/hqgkbkjr.html)
- Program Management (http://developer.novell.com/ndk/doc/clib/prog_enu/data/h9qu926c.html)
- NLM Threads Management (http://developer.novell.com/ndk/doc/clib/thmp_enu/data/h7g6q8vc.html)
- Connection, Message, and NCP Extensions (http://developer.novell.com/ndk/doc/clib/cmgnxenu/data/hvxfva0i.html)
- Multiple and Inter-File Services (http://developer.novell.com/ndk/doc/clib/mlti_enu/data/hby40vgi.html)
- Single and Intra-File Services (http://developer.novell.com/ndk/doc/clib/sngl_enu/data/h68b1qom.html)
- Client Management (http://developer.novell.com/ndk/doc/clib/clnt_enu/data/he77rked.html)

- Network Management (http://developer.novell.com/ndk/doc/clib/nwrk_enu/data/hvyko5s2.html)
- Server Management (http://developer.novell.com/ndk/doc/clib/srvr_enu/data/hzvjztxz.html)
- Internationalization (http://developer.novell.com/ndk/doc/clib/intl_enu/data/h70a28iu.html)
- Unicode (http://developer.novell.com/ndk/doc/clib/ucod_enu/data/hjg275fp.html)
- Sample Code (http://developer.novell.com/ndk/doc/clib/code_enu/data/hwtnc7wc.html)
- Getting Started with NetWare Cross-Platform Libraries for C (http://developer.novell.com/ndk/doc/clib/startenu/data/hv9aw5v8.html)
- Bindery Management (http://developer.novell.com/ndk/doc/clib/bind_enu/data/h9qzn7u5.html)

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

For help with CLib and XPlat problems or questions, visit the NLM and NetWare Libraries for C (including CLIB and XPlat) Developer Support Forums (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.

Concepts

1

This documentation describes NetWare volumes, their functions, and features.

1.1 Volume Basics

A NetWare volume is the highest level in the NetWare directory structure (the network equivalent of a DOS root directory). Volumes are divided into blocks made up of sectors. Each sector is 512 bytes. The default block size is 4 KB. (The number of blocks per volume depends on the size of the volume.)

A server running versions of NetWare before 3.11 can accommodate up to 32 volumes. A server running NetWare 3.11 or above can accommodate up to 64 volumes.

A NetWare server identifies volumes by name and number. Knowing either value allows you to find the other.

- NWGetVolumeNumber (page 31) uses the volume name to return the volume number.
- NWGetVolumeName (page 29) uses the volume number to find the volume name.

1.2 Volume Information Functions

These functions return information about a volume:

Function	Header File	Description
NWGetVolumeInfoWithHandle	nwvol.h	Returns information for the volume on which the specified directory is found.
NWGetVolumeInfoWithNumber	nwvol.h	Returns volume information for the specified volume.
NWGetVolumeName	nwvol.h	Returns the name of the volume associated with the specified volume number.
NWGetVolumeNumber	nwvol.h	Returns the volume number based on the NetWare® server connection ID and volume name.
NWGetExtendedVolumeInfo	nwvol.h	Returns extended information for the specified volume.

1.3 Volume Utilization and Restriction Functions

These functions access space restrictions and utilization statistics for a volume.

Function	Header File	Description
NWGetDiskUtilization	nwvol.h	Returns disk usage for a specified bindery object on a volume.

Function	Header File	Description
NWGetObjDiskRestrictions	nwvol.h	Returns the restriction on a volume for the specified bindery object.
NWRemoveObjectDiskRestrictions	nwvol.h	Removes all disk restrictions for the specified object on a volume.
NWScanVolDiskRestrictions2	nwvol.h	Returns a list of objects and their disk restrictions on a volume.
NWSetObjectVolSpaceLimit	nwvol.h	Adds a user disk space restriction to a volume.

Tasks

2

This documentation describes common tasks associated with the volume functions.

2.1 Reading Volume Information

NetWare® volume information indicates the amount of space available on a volume. It includes the block size (number of sectors per block) and the following totals:

- Total blocks available
- Total blocks in use
- Total directory entries available
- Total directory entries in use

It also indicates whether the volume is removable.

Two functions enable you to read volume information, one by means of volume number and the other by directory handle:

- NWGetVolumeInfoWithNumber (page 26) takes a volume number.
- NWGetVolumeInfoWithHandle (page 23) takes a directory handle.

Additional volume information is available at the directory level for 3.11 and above by calling NWGetDirSpaceInfo (Multiple and Inter-File Services). For example: block and directory entry totals and statistics for purgeable blocks.

2.2 Managing Disk Space

With NetWare® 3.11 and above, you can control the total amount of space available to each object within a volume.

NetWare 3.11 and above servers let you restrict the number of 4 KB blocks available to a specified object. One function sets disk space restrictions and two functions read restrictions:

- NWSetObjectVolSpaceLimit (page 40) sets an object's disk space restriction in blocks. On NetWare 4.x and above servers, the restriction can range from 0 to 0x08000000. On 3.11 servers, the range is from 0 to 0x40000000.
- NWGetObjDiskRestrictions (page 21) returns the restriction for a specified object.
- NWScanVolDiskRestrictions2 (page 35) can be called iteratively to build a list of objects that are assigned disk space restrictions.

  To remove restrictions for a specific object on a volume, call NWRemoveObjectDiskRestrictions (page 33).
- NWGetDiskUtilization (page 16) returns the number of files, directories, and blocks an object is using on a volume.

Functions

3

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

NWGetDiskUtilization

Allows a client to determine how much physical space the specified object ID is using on the given 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: Volume

Syntax

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

N_EXTERN_LIBRARY( NWCCODE ) NWGetDiskUtilization  (
   NWCONN_HANDLE   conn,
   nuint32         objID,
   nuint8          volNum,
   pnuint16        usedDirectories,
   pnuint16        usedFiles,
   pnuint16        usedBlocks);
```

Pascal Syntax

```
uses calwin32

Function NWGetDiskUtilization
  (conn : NWCONN_HANDLE;
   objID : nuint32;
   volNum : nuint8;
   usedDirectories : pnuint16;
   usedFiles : pnuint16;
   usedBlocks : pnuint16
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare® server connection handle.

objID

(IN) Specifies the object ID.

volNum

(IN) Specifies the volume number.

usedDirectories

(OUT) Points to the number of directories on the volume owned by `objID`.

usedFiles

(OUT) Points to the number of files on the volume owned by `objID`.

usedBlocks

(OUT) Points to the number of physical volume blocks occupied by files owned by `objID`.

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
0x89A1	DIRECTORY_IO_ERROR
0x89F2	NO_OBJECT_READ_PRIVILEGE
0x89FC	NO_SUCH_OBJECT

Remarks

`usedBlocks` will return incorrect information for disks larger than 268 megabytes. Call NWGetObjDiskRestrictions to get the disk space being used by an object.

Clients who are SUPERVISOR equivalent can call NWGetDiskUtilization for any object. Clients not having SUPERVISOR rights can call NWGetDiskUtilization only for the object used when logging in.

Call either NWGetObjectID or NWDSMapNameToID to get the object ID.

NWGetDiskUtilization will not validate `objID`. If `objID` is invalid or does not exist on the server, NWGetDiskUtilization will return zero (0) for the disk utilization.

NCP Calls

0x2222 23 14 Get Disk Utilization
0x2222 23 54 Get Object Name

See Also

NWDSMapNameToID (*NDS Core Services*), NWGetObjDiskRestrictions (page 21), NWGetObjectID (*NDK: Bindery Management*)

NWGetExtendedVolumeInfo

Returns extended volume information

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: Volume

Syntax

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

N_EXTERN_LIBRARY( NWCCODE ) NWGetExtendedVolumeInfo  (
   NWCONN_HANDLE              conn,
   nuint16                    volNum,
   NWVolExtendedInfo N_FAR   *volInfo);
```

Pascal Syntax

```
uses calwin32

Function NWGetExtendedVolumeInfo
  (conn : NWCONN_HANDLE;
   volNum : nuint16;
   Var volInfo : NWVolExtendedInfo
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNum

(IN) Specifies the volume number.

volInfo

(OUT) Points to NWVolExtendedInfo, which receives information.

Return Values

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

0x0000	SUCCESSFUL
0x8998	VOLUME_DOES_NOT_EXIST
0x897E	NCP_BOUNDARY_CHECK_FAILED
0x89FB	NO_SUCH_PROPERTY

Remarks

NWGetExtendedVolumeInfo returns information based on the volume block size (64 KB), which can be determined using the formula:

```
(sectorSize*sectorsPerCluster)/1024
```

NWGetExtendedVolumeInfo must be called for a licensed connection or NO_SUCH_PROPERTY will be returned.

For sample code, see Developer Q&A (http://support.novell.com/techcenter/qna/dnq20030204.html).

NCP Calls

0x2222 22 51 Get Extended Volume Information

NWGetObjDiskRestrictions

Returns the disk restrictions imposed on an object for the specified volume number

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: Volume

Syntax

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

N_EXTERN_LIBRARY( NWCCODE ) NWGetObjDiskRestrictions  (
   NWCONN_HANDLE   conn,
   nuint8          volNumber,
   nuint32         objectID,
   pnuint32        restriction,
   pnuint32        inUse);
```

Pascal Syntax

```
uses calwin32

Function NWGetObjDiskRestrictions
  (conn : NWCONN_HANDLE;
   volNumber : nuint8;
   objectID : nuint32;
   restriction : pnuint32;
   inUse : pnuint32
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNumber

(IN) Specifies the volume number for which to return the restrictions.

objectID

(IN) Specifies the object ID.

restriction

(OUT) Points to the buffer containing the number of blocks the object can use.

inUse

(OUT) Points to the buffer containing the number of blocks the object is currently using.

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

Remarks

The restrictions are returned in units of 4KB blocks and ignore the block size of the volume.

NOTE: If the restriction equals 0x40000000, the object has no restrictions.

NCP Calls

0x2222 22 41 Get Object Disk Usage And Restrictions

See Also

NWGetExtendedVolumeInfo (page 19), NWSetObjectVolSpaceLimit (page 40)

NWGetVolumeInfoWithHandle

Returns the physical information or data of a server's volumes

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: Volume

Syntax

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

N_EXTERN_LIBRARY(NWCCODE) NWGetVolumeInfoWithHandle (
   NWCONN_HANDLE   conn,
   NWDIR_HANDLE    dirHandle,
   pnstr8          volName,
   pnuint16        totalBlocks,
   pnuint16        sectorsPerBlock,
   pnuint16        availableBlocks,
   pnuint16        totalDirEntries,
   pnuint16        availableDirEntries,
   pnuint16        volIsRemovableFlag);
```

Pascal Syntax

```
uses calwin32

Function NWGetVolumeInfoWithHandle(
   conn : NWCONN_HANDLE;
   dirHandle : NWDIR_HANDLE;
   volName : pnstr8;
   totalBlocks : pnuint16;
   sectorsPerBlock : pnuint16;
   availableBlocks : pnuint16;
   totalDirEntries : pnuint16;
   availableDirEntries : pnuint16;
   volIsRemovableFlag : pnuint16
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

dirHandle

(IN) Specifies the directory handle pointing to the directory on the volume whose information is to be reported.

volName

(OUT) Points to the volume name (optional 17 character buffer including the terminating NULL).

totalBlocks

(OUT) Points to the total number of blocks on the volume (optional).

sectorsPerBlock

(OUT) Points to the number of sectors per block (optional).

availableBlocks

(OUT) Points to the total number of unused blocks on the volume (optional).

totalDirEntries

(OUT) Points to the total number of physical directory entries (optional).

availableDirEntries

(OUT) Points to the number of unused directory entries (optional).

volIsRemovableFlag

(OUT) Set to NULL. The value in this field is never set.

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
0x89FF	HARDWARE_FAILURE

Remarks

NWGetVolumeInfoWithHandle returns a 16-bit number in the `totalBlocks` parameter. If the volume size is greater than a 16-bit number (or 256 megabytes), NWGetDirSpaceInfo should be called.

`dirHandle` is an index number (1 through 255) pointing to a volume, directory, or subdirectory on the NetWare server. Directory handles are recorded in the Directory Handle Table maintained by the server for each logged-in workstation. When a workstation allocates a directory handle, the NetWare server enters the volume number and directory entry number for the specified directory into the Directory Handle Table. Applications running on the workstation can then refer to a directory using a directory handle, which is actually an index into the Directory Handle Table.

Since all of the output parameters are optional, substitute NULL for unwanted information. However, all parameter positions must be filled.

Volumes use logical sector sizes of 512 bytes. If the physical media uses a different sector size, the server performs appropriate mappings. Volume space is allocated in groups of sectors called blocks.

`sectorsPerBlock` indicates how many 512-byte sectors are contained in each block of the specified volume.

`totalDirEntries` indicates how many directory entries were allocated for the specified volume during installation. If this information is meaningless under a given server's implementation, it is 0xFFFF.

`volIsRemovableFlag` indicates whether a user can physically remove the volume from the NetWare server. It returns one of the following values:

```
0x0000 = not removable/fixed media
non-zero = removable/mountable
```

With NetWare 4.x, 5.x, 6.x, and SFTIII, the volume sector size can be changed from the 512-byte default. If changed, NWGetVolumeInfoWithHandle may return adjusted data meeting DOS requirements. `totalBlocks`, `sectorsPerBlock` and `availableBlocks` may be affected. To see the actual field size, call NWGetExtendedVolumeInfo.

NOTE: Block size can be found by calling NWGetExtendedVolumeInfo and multiplying `sectorSize` and `sectorPerCluster`.

NCP Calls

0x2222 22 21 Get Volume Info With Handle

See Also

NWGetDirSpaceInfo (Multiple and Inter-File Services), NWGetVolumeInfoWithNumber (page 26)

NWGetVolumeInfoWithNumber

Returns information for the specified volume by passing a volume number, allowing a client to check the physical space available on 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: Volume

Syntax

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

N_EXTERN_LIBRARY( NWCCODE ) NWGetVolumeInfoWithNumber  (
   NWCONN_HANDLE   conn,
   nuint16         volNum,
   pnstr8          volName,
   pnuint16        totalBlocks,
   pnuint16        sectorsPerBlock,
   pnuint16        availableBlocks,
   pnuint16        totalDirEntries,
   pnuint16        availableDirEntries,
   pnuint16        volIsRemovableFlag);
```

Pascal Syntax

```
uses calwin32

Function NWGetVolumeInfoWithNumber
  (conn : NWCONN_HANDLE;
   volNum : nuint16;
   volName : pnstr8;
   totalBlocks : pnuint16;
   sectorsPerBlock : pnuint16;
   availableBlocks : pnuint16;
   totalDirEntries : pnuint16;
   availableDirEntries : pnuint16;
   volIsRemovableFlag : pnuint16
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNum

(IN) Specifies the volume number of the volume for which information is being obtained.

volName

(OUT) Points to the volume name (optional 17 character buffer including the terminating NULL).

totalBlocks

(OUT) Points to the total number of blocks on the volume (optional).

sectorsPerBlock

(OUT) Points to the number of sectors per block (optional).

availableBlocks

(OUT) Points to the number of unused blocks on the volume (optional).

totalDirEntries

(OUT) Points to the total number of physical directory entries (optional).

availableDirEntries

(OUT) Points to the number of unused directory entries (optional).

volIsRemovableFlag

(OUT) Set to NULL. The value in this field is never set.

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

Remarks

NWGetVolumeInfoWithNumber returns a 16-bit number in the `totalBlocks` parameter. If the volume size is greater than a 16-bit number (or 256 megabytes), NWGetDirSpaceInfo should be called.

`volNum` identifies the volume name on the NetWare server's Volume Table.

Volumes use logical sector sizes of 512 bytes. If the physical media uses a different sector size, the server performs appropriate mappings. Volume space is allocated in groups of sectors called blocks.

`sectorsPerBlock` indicates the number of 512-byte sectors contained in each block of the specified volume.

`totalDirEntries` indicates how many directory entries were allocated for the specified volume during installation. If this information is meaningless under a given server's implementation, it is 0xFFFF.

Since all of the output parameters are optional, substitute a NULL for unwanted information. However, all parameter positions must be filled.

With NetWare 4.x, 5.x, 6.x, and SFTIII, the volume sector size can be changed from the 512-byte default. If changed, NWGetVolumeInfoWithHandle may return adjusted data that meets DOS requirements. `totalBlocks`, `sectorsPerBlock`, and `availableBlocks` may be affected. To see the actual field size, call NWGetExtendedVolumeInfo.

NOTE: Block size can be found by calling NWGetExtendedVolumeInfo and multiplying `sectorSize` and `sectorPerCluster`.

NCP Calls

0x2222 18 Get Volume Info With Number

See Also

NWGetDirSpaceInfo (Multiple and Inter-File Services), NWGetVolumeInfoWithHandle (page 23)

NWGetVolumeName

Returns the name of the volume associated with the specified volume number and 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: Volume

Syntax

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

N_EXTERN_LIBRARY( NWCCODE ) NWGetVolumeName  (
   NWCONN_HANDLE   conn,
   nuint16         volNum,
   pnstr8          volName);
```

Pascal Syntax

```
uses calwin32

Function NWGetVolumeName
  (conn : NWCONN_HANDLE;
   volNum : nuint16;
   volName : pnstr8
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNum

(IN) Specifies the volume number of the volume for which information is being obtained.

volName

(OUT) Points to the volume name (17 characters including the terminating NULL).

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
0x89FF	HARDWARE_FAILURE

Remarks

`volNum` identifies the volume name on the NetWare server's Volume Table. `volNum` needs to be between 0 and the maximum allowable volumes on the server.

NWGetVolumeName can be called to determine all volume numbers and volume names currently mounted on the specified NetWare server:

- For regular volumes, start the scan with volume number 0 and scan upwards.
- For clustered volumes, start the scan with volume number 255 and scan downwards.

SUCCESSFUL will be returned for each allowable volume number whether or not that volume exists on the specified server. For example, NetWare 3.x and above supports 64 volumes on each server. Calling NWGetVolumeName on each of the 64 volumes will return SUCCESSFUL even though the volume is not mounted.

NCP Calls

0x2222 22 6 Get Volume Name

See Also

NWGetVolumeNumber (page 31)

NWGetVolumeNumber

Returns the volume number based on the NetWare server connection handle and the volume 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: Volume

Syntax

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

N_EXTERN_LIBRARY(NWCCODE) NWGetVolumeNumber (
   NWCONN_HANDLE       conn,
   const nstr8 N_FAR  *volName,
   pnuint16            volNum);
```

Pascal Syntax

```
uses calwin32

Function NWGetVolumeNumber
  (conn : NWCONN_HANDLE;
   const volName : pnstr8;
   volNum : pnuint16
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volName

(IN) Points to the volume name (17 characters including the terminating NULL).

volNum

(OUT) Points to the volume number (identifies the volume on the NetWare server's Volume Table).

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

Remarks

For sample code, see Developer Q&A (http://support.novell.com/techcenter/qna/dnq20030204.html).

NCP Calls

0x2222 22 5 Get Volume Number

See Also

NWGetVolumeName (page 29), NWGetVolumeInfoWithNumber (page 26)

NWRemoveObjectDiskRestrictions

Removes any disk restrictions for the specified object, for the specified volume, 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: Volume

Syntax

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

N_EXTERN_LIBRARY( NWCCODE ) NWRemoveObjectDiskRestrictions  (
   NWCONN_HANDLE   conn,
   nuint8          volNum,
   nuint32         objID);
```

Pascal Syntax

```
uses calwin32

Function NWRemoveObjectDiskRestrictions
  (conn : NWCONN_HANDLE;
   volNum : nuint8;
   objID : nuint32
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNum

(IN) Specifies the volume number for which to remove restrictions.

objID

(IN) Specifies the object ID for which to remove restrictions.

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
0x8998	VOLUME_DOES_NOT_EXIST
0x89FE	NetWare Error (object has no restrictions)

NCP Calls

0x2222 22 34 Remove User Disk Space Restriction

NWScanVolDiskRestrictions2

Returns a list of the disk restrictions 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: Volume

Syntax

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

N_EXTERN_LIBRARY( NWCCODE ) NWScanVolDiskRestrictions2   (
   NWCONN_HANDLE                conn,
   nuint8                       volNum,
   pnuint32                     iterHnd,
   NWVOL_RESTRICTIONS  N_FAR   *volInfo);
```

Pascal Syntax

```
uses calwin32

Function NWScanVolDiskRestrictions2
  (conn : NWCONN_HANDLE;
   volNum : nuint8;
   iterhandle : pnuint32;
   Var volInfo : NWVOL_RESTRICTIONS
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNum

(IN) Specifies the volume number for which to return the restrictions.

iterHnd

(OUT) Points to the sequence number to use in the search (set to 0 initially).

volInfo

(OUT) Points to NWVOL_RESTRICTIONS.

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

Remarks

NWScanVolDiskRestrictions2 replaces NWScanVolDiskRestrictions. The new function uses a larger structure for the volume restrictions that allows up to 16 restrictions per volume.

NOTE: Calling NWScanVolDiskRestrictions when you have more than 12 restrictions per volume causes random failures. For this reason, call NWScanVolDiskRestrictions2 exclusively.

The information returned in NWVOL_RESTRICTIONS contains the object restrictions that have been made for the volume. All restrictions are returned in 4K blocks. If the restriction is greater than 0x40000000 on a 3.1 server or 0x80000000 on a 4.x and above server, the object has no restrictions.

IMPORTANT: NWScanVolDiskRestrictions2 is called iteratively to retrieve information on all disk space restrictions. The number of entries is returned in the `volInfo.numberOfEntries` field. This value must be added to the previous `iterHnd` to obtain the value for the next iterative call.

NCP Calls

0x2222 22 32 Scan Volume's User Disk Restrictions

NWScanMountedVolumeList

Returns a list of mounted volumes

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: Volume

Syntax

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

NWCCODE  NWScanMountedVolumeList (
   nuint32                        conn,
   nuint32                        volRequestFlags,
   nuint32                        nameSpace,
   pnuint32                       iterHandle,
   nuint32                        numberItems,
   pnuint32                       numberReturned,
   NWVolMountNumWithName N_FAR   *volInfo);
```

Pascal Syntax

```
Function NWScanMountedVolumeList (
   conn : nuint32;
   volRequestFlags : nuint32;
   nameSpace : nuint32;
   VAR iterHandle : nuint32;
   numberItems : nuint32;
   VAR numberReturned : nuint32;
   volInfo : pNWVolMountNumWithName
) : NWCCODE;  stdcall;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volRequestFlags

(IN) Specifies only the volume number or the volume number with the volume name.

nameSpace

(IN) Specifies the name space for which you want to get the mounted volume list.

iterHandle

(IN/OUT) Points to an nuint32 containing the number of the next record to be scanned. (Set to 0 for the first call.)

numberItems

(IN) Specifies the size of the array passed into `volMountedArr`.

numberReturned

(OUT) Specifies how many volumes are actually in the array pointed to by `volMountedArr`.

volMountArr

(OUT) Points to an array of NWVolMountNumWithName structures containg a list of volumes returned from the current call.

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

Remarks

NWScanMountedVolumeList allows you to pass in a pointer to a variably sized array of NWVolMountNumWithName structures. On return, that pointer points to a list of mounted vloumes. Based on the size of the array and number of mounted volumes, NWScanMountedVolumeList might return the complete list in only one call or might take multiple calls.

To call NWScanMountedVolumeList iteratively, pass in zero for the `iterHandle` parameter on the first call. On return, check `iterHandle` to get the number of the next record to scan for mounted volumes. When `iterHandle` contains zero on return, there are no more records to scan.

The `volRequestFlags` parameter can take one of the following values:

NW_VOLUME_NUMBER_ONLY	0
NW_VOLUME_NUMBER_AND_NAME	1

The `nameSpace` parameter can use any of the constant values identified in Naming Conventions (Multiple and Inter-File Services).

NWScanMountedVolumeList is implented through a call to NCP 0x2222 22 52. This NCP is supported on NetWare 4.x and above.

NCP Calls

0x2222 22 52 Get Mount Volume List

NWSetObjectVolSpaceLimit

Sets an object's disk space limit on 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: Volume

Syntax

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

N_EXTERN_LIBRARY( NWCCODE ) NWSetObjectVolSpaceLimit  (
   NWCONN_HANDLE   conn,
   nuint16         volNum,
   nuint32         objID,
   nuint32         restriction);
```

Pascal Syntax

```
uses calwin32

Function NWSetObjectVolSpaceLimit
  (conn : NWCONN_HANDLE;
   volNum : nuint16;
   objID : nuint32;
   restriction : nuint32
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

volNum

(IN) Specifies the volume number for which to set the space limit.

objID

(IN) Specifies the object ID for which to limit the volume space.

restriction

(IN) Specifies the number of blocks (in 4KB sizes) to limit the volume 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
0x8996	SERVER_OUT_OF_MEMORY
0x8998	VOLUME_DOES_NOT_EXIST

Remarks

The restrictions are returned in units of 4K blocks.

NOTE: If the restriction equals 0x40000000, the object has no restrictions.

NCP Calls

0x2222 22 33 Add User Disk Space Restriction

See Also

NWGetExtendedVolumeInfo (page 19), NWGetObjDiskRestrictions (page 21)

Structures

4

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

NWOBJ_REST

Contains an object ID with the restrictions placed on the object for a certain volume (to be used with NWVOL_RESTRICTIONS)

Service: Volume

Defined In: nwvol.h

Structure

```
typedef struct
{
   nuint32   objectID ;
   nuint32   restriction ;
} NWOBJ_REST;
```

Pascal Structure

```
uses calwin32

  NWOBJ_REST = packed Record
    objectID : nuint32;
    restriction : nuint32;
  End;
```

Fields

objectID

Specifies the NDS ID for an object.

restriction

Specifies by the number of blocks, the amount of restriction placed on the object.

NWVolExtendedInfo

Contains extended information for a volume

Service: Volume

Defined In: nwvol.h

Structure

```
typedef struct {
   nuint32   volType ;
   nuint32   statusFlag ;
   nuint32   sectorSize ;
   nuint32   sectorsPerCluster ;
   nuint32   volSizeInClusters ;
   nuint32   freeClusters ;
   nuint32   subAllocFreeableClusters ;
   nuint32   freeableLimboSectors ;
   nuint32   nonfreeableLimboSectors ;
   nuint32   availSubAllocSectors ;
   nuint32   nonuseableSubAllocSectors ;
   nuint32   subAllocClusters ;
   nuint32   numDataStreams ;
   nuint32   numLimboDataStreams ;
   nuint32   oldestDelFileAgeInTicks ;
   nuint32   numCompressedDataStreams ;
   nuint32   numCompressedLimboDataStreams ;
   nuint32   numNoncompressibleDataStreams ;
   nuint32   precompressedSectors ;
   nuint32   compressedSectors ;
   nuint32   numMigratedDataStreams ;
   nuint32   migratedSectors ;
   nuint32   clustersUsedByFAT ;
   nuint32   clustersUsedByDirs ;
   nuint32   clustersUsedByExtDirs ;
   nuint32   totalDirEntries ;
   nuint32   unusedDirEntries ;
   nuint32   totalExtDirExtants ;
   nuint32   unusedExtDirExtants ;
   nuint32   extAttrsDefined ;
   nuint32   extAttrExtantsUsed ;
   nuint32   DirectoryServicesObjectID ;
   nuint32   volLastModifiedDateAndTime ;
} NWVolExtendedInfo;
```

Pascal Structure

```
uses calwin32

NWVolExtendedInfo = packed Record
    volType : nuint32;
```

```
    statusFlag : nuint32;
    sectorSize : nuint32;
    sectorsPerCluster : nuint32;
    volSizeInClusters : nuint32;
    freeClusters : nuint32;
    subAllocFreeableClusters : nuint32;
    freeableLimboSectors : nuint32;
    nonfreeableLimboSectors : nuint32;
    availSubAllocSectors : nuint32;
    nonuseableSubAllocSectors : nuint32;
    subAllocClusters : nuint32;
    numDataStreams : nuint32;
    numLimboDataStreams : nuint32;
    oldestDelFileAgeInTicks : nuint32;
    numCompressedDataStreams : nuint32;
    numCompressedLimboDataStreams : nuint32;
    numNoncompressibleDataStreams : nuint32;
    precompressedSectors : nuint32;
    compressedSectors : nuint32;
    numMigratedDataStreams : nuint32;
    migratedSectors : nuint32;
    clustersUsedByFAT : nuint32;
    clustersUsedByDirs : nuint32;
    clustersUsedByExtDirs : nuint32;
    totalDirEntries : nuint32;
    unusedDirEntries : nuint32;
    totalExtDirExtants : nuint32;
    unusedExtDirExtants : nuint32;
    extAttrsDefined : nuint32;
    extAttrExtantsUsed : nuint32;
    DirectoryServicesObjectID : nuint32;
    volLastModifiedDateAndTime : nuint32;
  End;
```

Fields

volType

Specifies different volumes that may be supported in the future.

statusFlag

Specifies the options currently available in this volume:

C Value	Pascal Value	Value Name
0x01	$01	NWSubAllocEnableBit
0x02	$02	NWCompressionEnabledBit
0x04	$04	NWMigrationEnableBit
0x08	$08	NWAuditingEnabledBit
0x10	$10	NWReadOnlyEnableBit

C Value	Pascal Value	Value Name
0x80000000	$80000000	NWPSSEnabledBit—the volume is an NSS volume.

sectorSize

Specifies the sector size in bytes.

sectorsPerCluster

Specifies the number of sectors per cluster.

volSizeInClusters

Specifies the size, in clusters, of the volume.

freeClusters

Specifies the number of clusters currently free for allocation. This does not include space currently available from deleted (limbo) files, nor space that could be reclaimed from the suballocation file system.

subAllocFreeableClusters

Specifies the space that could be reclaimed from the suballocation file system.

freeableLimboSectors

Specifies the disk space, in sectors, that could be freed from deleted files.

nonfreeableLimboSectors

Specifies the disk space, in sectors, that are currently in deleted files and not aged enough to be classified as `FreeableLimboClusters`. These will be migrated to the status of `FreeableLimboCluster` after time.

availSubAllocSectors

Specifies the space available to the suballocation file system, but not freeable to return as sectors.

nonuseableSubAllocSectors

Specifies the disk space wasted by the suballocation file system. These sectors cannot be allocated by the suballocation system or used as regular sectors.

subAllocClusters

Specifies the disk space being used by the suballocation file system.

numDataStreams

Specifies the number of data streams for real files with data allocated to them.

numLimboDataStreams

Specifies the number of data streams for deleted files with data allocated to them.

oldestDelFileAgeInTicks

Specifies the current age of the oldest file in ticks.

numCompressedDataStreams

Specifies the number of data streams for compressed real files.

numCompressedLimboDataStreams

Specifies the count of data streams for compressed deleted files.

numNoncompressibleDataStreams

Specifies the data streams found not compressable (real and deleted).

precompressedSectors

Specifies the disk space allocated to all files before they were compressed (includes "hole" space).

compressedSectors

Specifies the disk space used by all compressed files.

numMigratedDataStreams

Specifies the number of migrated data streams.

migratedSectors

Specifies the migrated disk space (in sectors).

clustersUsedByFAT

Specifies the disk space (in clusters) that is used by the FAT table.

clustersUsedByDirs

Specifies the disk space (in clusters) that is used by directories.

clustersUsedByExtDirs

Specifies the disk space (in clusters) that is used by the extended directory space.

totalDirEntries

Specifies the total number of directories available on the volume.

unusedDirEntries

Specifies the total directory entries unused on volume.

totalExtDirExtants

Specifies the amount of extended directory space extants (128 bytes each) that are available on the volume.

unusedExtDirExtants

Specifies the amount of extended directory space extants (128 bytes each) that are unused on the volume.

extAttrsDefined

Specifies the number of extended attributes that are defined on the volume.

extAttrExtantsUsed

Specifies the number of extended directory extants that are used by the extended attributes.

DirectoryServicesObjectID

Specifies the NDS ID for volume.

volLastModifiedDateAndTime

Specifies the last time any file or subdirectory within the volume was modified (tracked by the OS).

Remarks

The `volType` parameter can have the following values:

0 VlNetWare386
1 VlNetWare286
2 VlNetWare386v30
3 VlNetWare386v31

NWVolMountNumWithName

Returns the volume information.

Service: Volume

Defined In: nwvol.h

Structure

```
typedef struct NWVolMountNumWithName_tag
{
   nuint32   volumeNumber;
   nstr8     volumeName[NW_MAX_VOLUME_NAME_LEN];
} NWVolMountNumWithName;
```

Pascal Syntax

```
TYPE
   NWVolMountNumWithName  = packed RECORD
      volumeNumber : nuint32;
      volumeName : Array[1..NW_MAX_VOLUME_NAME_LEN]
         of nstr8;
      filler : Array[1..3] of nuint8;
 end;

pNWVolMountNumWithName = ^NWVolMountNumWithName;
```

Fields

`volumeNumber`

Specifies the number of the volume.

`volumeName`

Specifies the volume name.

NWVOL_RESTRICTIONS

Returns a list of objects with space restrictions on a volume

Service: Volume

Defined In: nwvol.h

Structure

```
typedef struct
{
   nuint8   numberOfEntries;
   struct
   {
      nuint32   objectID;
      nuint32   restriction;
   } resInfo[16];
} NWVOL_RESTRICTIONS;
```

Pascal Structure

```
uses calwin32

  NWVOL_RESTRICTIONS = packed Record
    numberOfEntries : nuint8;
    resInfo : Array[0..15] Of RES_INFO;
  End;

  RES_INFO = Record
    objectID : nuint32;
    restriction : nuint32;
  End;
```

Fields

`numberOfEntries`

Specifies the number of objects in the list (0-16 objects).

`objectID`

Specifies the ID of the NDS object (in Hi-Lo format). This value needs to be byte swapped when passed to NWGetObjectName or NWDSMapIDToName.

`restriction`

Specifies the size, in 4KB blocks, of the restriction placed on an object (Lo-Hi format).

VOL_STATS

Contains volume statistics

Service: Volume

Defined In: nwvol.h

Structure

```
typedef struct
{
   nint32    systemElapsedTime ;
   nuint8    volumeNumber ;
   nuint8    logicalDriveNumber ;
   nuint16   sectorsPerBlock ;
   nuint16   startingBlock ;
   nuint16   totalBlocks ;
   nuint16   availableBlocks ;
   nuint16   totalDirectorySlots ;
   nuint16   availableDirectorySlots ;
   nuint16   maxDirectorySlotsUsed ;
   nuint8    isHashing ;
   nuint8    isCaching ;
   nuint8    isRemovable ;
   nuint8    isMounted ;
   nstr8     volumeName [16];
} VOL_STATS;
```

Pascal Structure

```
uses calwin32

VOL_STATS = packed Record
    systemElapsedTime : nint32;
    volumeNumber : nuint8;
    logicalDriveNumber : nuint8;
    sectorsPerBlock : nuint16;
    startingBlock : nuint16;
    totalBlocks : nuint16;
    availableBlocks : nuint16;
    totalDirectorySlots : nuint16;
    availableDirectorySlots : nuint16;
    maxDirectorySlotsUsed : nuint16;
    isHashing : nuint8;
    isCaching : nuint8;
    isRemovable : nuint8;
    isMounted : nuint8;
    volumeName : Array[0..15] Of nstr8;
  End;
```

Fields

systemElapsedTime

Specifies how long the server has been up. This value is returned in ticks (units of approximately 1/18 second) and is used to determine the amount of time elapsing between consecutive calls. After reaching a value of 0xFFFFFFFF, the value wraps back to zero.

volumeNumber

Specifies the number of a volume in a volume table on a server. SYS volume is always zero.

logicalDriveNumber

Specifies the logical drive number of the drive on which the volume exists.

sectorsPerBlock

Specifies the number of 512-byte sectors contained in each block of the specified volume. NWU (NetWare for Unix) does not support this field and returns a zero.

startingBlock

Specifies the number of the first block of the volume.

totalBlocks

Specifies the number of blocks in the specified volume. NWU (NetWare for Unix) returns the total amount of disk space on the volume's host file system. All volumes mounted from the same file system will return the same value.

availableBlocks

Specifies the number of unused blocks in the specified volume. NWU (NetWare for Unix) returns the total amount of disk space on the host file system. All volumes mounted from the same file system will return the same value.

totalDirectorySlots

Specifies the number of directory slots allocated for the specified volume. NWU (NetWare for Unix) returns the number of files that can be created to track NetWare file and trustee information.

availableDirectorySlots

Specifies the number of directories that can be created, based on the differences between the total allowable number of directories and the number of directories already created. NWU (NetWare for Unix) returns the number of directories that can be created.

maxDirectorySlotsUsed

Specifies the greatest number of directory slots ever used at one time on the volume. NWU (NetWare for Unix) does not support this field and returns a zero.

isHashing

Specifies whether the volume is hashing in server memory (0=not hashing). Only NetWare 2.x servers return a valid value.

isCaching

Specifies whether the volume is caching in server memory (0=volume not caching). Only NetWare 2.x servers returns a valid value.

isRemovable

Specifies if a user can physically remove the volume from the server (0=cannot be removed). Only NetWare 3.x and 4.x servers return a valid value.

isMounted

Specifies whether the volume is physically mounted in the server (0=volume is not mounted). Only NetWare 3.x and 4.x servers return a valid value.

volumeName

Specifies the name given to the volume (1 to 16 characters long). It cannot contain asterisks (*), question marks (?), colons (:), slashes (/), or backslashes (\). If the name is less than 16 characters, the remaining characters must be null. NWU (NetWare for Unix) returns the NetWare name for the volume.

Server-Based Volume Management Functions

5

This documentation alphabetically lists the server-based volume management functions and describes their purpose, syntax, parameters, and return values.

GetNumberOfVolumes

Returns the number of volumes for the local server

Local Servers: blocking

Remote Servers: N/A

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

Platform: NLM

SMP Aware: No

Service: Volume

Syntax

```
#include <nwdir.h>

 LONG GetNumberOfVolumes (void);
```

Return Values

This function returns the number of volumes for the local server.

Remarks

This returns the number of volumes currently mounted on the local server.

See Also

GetVolumeInformation (page 57)

Example

```
#include <stdlib.h>
#include <nwdir.h>

 printf("Number of volumes on local server = %d\n",
         GetNumberOfVolumes() );
```

GetVolumeInformation

Returns information about a volume

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 GetVolumeInformation (
    WORD            fileServerID,
    BYTE            volumeNumber,
    int             structSize,
    VOLUME_STATS   *volumeStatistics);
```

Parameters

`fileServerID`

(IN) 0 = Local server.

`volumeNumber`

(IN) Specifies the volume number to return information on.

`structSize`

(IN) Specifies the size (in bytes) of the information to return in `volumeStatistics` .

`volumeStatistics`

(OUT) Receives information about the specified volume.

Return Values

Decimal	Hex	Constant
0	(0x00)	ESUCCESS
152	(0x98)	ERR_INVALID_VOLUME
NetWare Error		UNSUCCESSFUL

Remarks

If `structSize` is less than the size of VOLUME_STATS, then only the first `structSize` bytes of VOLUME_STATS are returned.

The VOLUME_STATS structure, pointed to by the `volumeStatistics` parameter, has the following format:

```
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   purgableBlocks;
LONG   notYetPurgableBlocks;
```

IMPORTANT: With large volumes, the number of blocks to be returned in `totalBlocks` or `availableBlocks` may be greater than 64K, resulting in inaccurate field values because of limited field size. In such instances, use GetVolumeStatistics instead of this function.

The *isRemovable* field always returns true.

See Also

GetVolumeInfoWithNumber (page 60), GetVolumeName (page 62), GetVolumeNumber (page 64), GetVolumeStatistics (page 66)

Example

```
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>
#include <fcntl.h>
#include <nwshare.h>
#include <nwdir.h>
#include <nwbitops.h>
#include <nwtts.h>
#include <nwbindry.h>
#include <time.h>

 main()
 {

    int            rc;
```

```
   VOLUME_STATS   vs;
   char
          svn[10];
   int            vn;

   printf("volume number: ");
   gets(svn);
   vn = atoi(svn);
   rc = GetVolumeInformation(0,vn,sizeof (vs), &vs);

if(rc)
   {
      printf("rc = %d\r\n",rc);
      printf("errno = %d\r\n",errno);
      printf("%s\r\n",strerror(errno));

   }
   else
   {
      printf("systemElapsedTime = %d\r\n",vs.systemElapsedTime);
      printf("volumeNumber = %d\r\n",vs.volumeNumber);
      printf("logicalDriveNumber = %d\r\n",vs.logicalDriveNumber);
      printf("sectorsPerBlock = %d\r\n",vs.sectorsPerBlock);
      printf("startingBlock = %d\r\n",vs.startingBlock);
      printf("totalBlocks = %d\r\n",vs.totalBlocks);

      printf("availableBlocks = %d\r\n",vs.availableBlocks);

      printf("totalDirectorySlots = %d\r\n",
              vs.totalDirectorySlots);
      printf("availableDirectorySlots = %d\r\n",
              vs.availableDirectorySlots);

      printf("maxDirectorySlotsUsed = %d\r\n",
              vs.maxDirectorySlotsUsed);
      printf("isHashing = %d\r\n",vs.isHashing);

      printf("isRemovable = %d\r\n",vs.isRemovable);

      printf("isMounted = %d\r\n",vs.isMounted);
      printf("volumeName = %s\r\n",vs.volumeName);
   }
}
```

GetVolumeInfoWithNumber

Returns information about a volume by volume number

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 GetVolumeInfoWithNumber (
    BYTE    volumeNumber,
    char   *volumeName,
    WORD   *totalBlocks,
    WORD   *sectorsPerBlock,
    WORD   *availableBlocks,
    WORD   *totalDirectorySlots,
    WORD   *availableDirectorySlots,
    WORD   *volumeIsRemovable);
```

Parameters

volumeNumber

(IN) Specifies the number of the volume slot (0-254 for NetWare 5.x and 6.x, 0-63 for NetWare 3.1 and 4x, 0-31 for previous versions). Even though the Volume Mount Table is 256 slots in size (0-255), the system reserves 255 as an invalid volume ID.

volumeName

(OUT) Returns a string containing the volume name (maximum 16 characters, including the NULL terminator).

totalBlocks

(OUT) Returns the number of blocks on the volume.

sectorsPerBlock

(OUT) Returns the number of sectors in a block.

availableBlocks

(OUT) Returns the number of unused blocks on the volume.

totalDirectorySlots

(OUT) Returns the number of directory slots on the volume.

availableDirectorySlots

(OUT) Returns the number of unused directory slots on the volume.

volumeIsRemovable

(OUT) Set to NULL. Always returns TRUE.

Return Values

Decimal	Hex	Constant
0	(0x00)	ESUCCESS
NetWare Error		UNSUCCESSFUL

Remarks

The GetVolumeInfoWithNumber function returns information about a volume by passing a volume number. The `volumeNumber` identifies the volume in the server's Volume Table. The Volume Table contains information about each volume on the server. A server running NetWare 3.1 or above can accommodate up to 64 volumes.

The `volumeName` parameter must be 16 bytes long. A volume name can be from 2 to 15 characters long plus the NULL terminator and cannot include spaces or the following characters:

*	Asterisk
?	Question mark
:	Colon
/	Slash
\	Backslash

The `sectorsPerBlock` parameter shows the number of 512-byte sectors contained in each block of the specified volume.

The `totalDirectorySlots` parameter shows the number of total directory slots available (NetWare 3.x and 4.x) or allocated (in the case of NetWare 2.x).

NOTE: This function requires console operator rights for NetWare 2.x, but not 3.x or 4.x. Therefore, if you call this function remotely on a NetWare 2.x server, you must have console operator rights.

See Also

GetVolumeInformation (page 57), GetVolumeName (page 62), GetVolumeNumber (page 64), GetVolumeStatistics (page 66)

GetVolumeName

Returns a volume name for a volume

Local Servers: nonblocking

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 GetVolumeName (
    int     volumeNumber,
    char   *volumeName);
```

Parameters

`volumeNumber`

(IN) Specifies the number of the volume slot (0-254 for NetWare 5.x and 6.x, 0-63 for NetWare 3.1 and 4x, 0-31 for previous versions). Even though the Volume Mount Table is 256 slots in size (0-255), the system reserves 255 as an invalid volume ID.

`volumeName`

(OUT) Points to the buffer in which to return the volume name (maximum 16 characters, including the NULL terminator).

Return Values

Decimal	Hex	Constant
0	(0x00)	ESUCCESS
152	(0x98)	ERR_VOLUME_DOES_NOT_EXIST

Remarks

The `volumeNumber` identifies the volume on the server's Volume Table, which contains information about each volume on the server.

If a volume *is* mounted in the referenced slot in the Volume Table, its name is returned in the `volumeName` parameter. If a volume is *not* mounted in that slot, the output parameter `volumeName` is NULL.

ESUCCESS is returned when the `volumeNumber` is valid, even if the volume is not mounted. In that case, you need to test for `volumeName` being valid.

An error is returned when an invalid `volumeNumber` is passed in.

See Also

GetVolumeInformation (page 57), GetVolumeInfoWithNumber (page 60), GetVolumeNumber (page 64), GetVolumeStatistics (page 66)

GetVolumeNumber

Returns the volume number for a volume

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 GetVolumeNumber (
    char   *volumeName,
    int    *volumeNumber);
```

Parameters

`volumeName`

(IN) Specifies the string containing the volume name (maximum 16 characters, including the NULL terminator).

`volumeNumber`

(OUT) Receives the volume number associated with the `volumeName` (0-254 for NetWare 5.x and 6.x, 0-63 for NetWare 3.1 and 4x, 0-31 for previous versions). Even though the Volume Mount Table is 256 slots in size (0-255), the system reserves 255 as an invalid volume ID.

Return Values

Decimal	Hex	Constant
0	(0x00)	ESUCCESS
152	(0x98)	ERR_VOLUME_DOES_NOT_EXIST

Remarks

The GetVolumeNumber function converts a volume name to a zero-based index. The `volumeName` parameter is 16 bytes long. A volume name can be from 2 to 16 characters long and cannot include spaces or the following characters:

*	Asterisk

?	Question Mark
:	Colon
/	Slash
\	Backslash

Wildcards are not allowed in the volume name.

The `volumeNumber` identifies the volume in the server's VolumeTable. The Volume Table contains information about each volume on the server. A server running NetWare 3.1 or above can accommodate up to 64 volumes.

See Also

GetVolumeInformation (page 57), GetVolumeInfoWithNumber (page 60), GetVolumeName (page 62), GetVolumeStatistics (page 66)

GetVolumeStatistics

Returns information about a volume

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 GetVolumeStatistics (
    WORD           fileServerID,
    BYTE           volumeNumber,
    int            structSize,
    VOLUME_INFO   *returnedVolumeStatistics);
```

Parameters

`fileServerID`

(IN) 0 = Local server.

`volumeNumber`

(IN) Specifies the volume number to return information on.

`structSize`

(IN) Specifies the size (in bytes) of the information to return in `volumeStatistics` .

`returnedVolumeStatistics`

(OUT) Receives information about the volume.

Return Values

Decimal	Hex	Constant
0	(0x00)	ESUCCESS
152	(0x98)	ERR_INVALID_VOLUME
NetWare Error		UNSUCCESSFUL

On remote calls, in the structure returned for `returnedVolumeStatistics`, GetVolumeStatistics returns -1 in the `systemElapsedTime` field and -1 in the `startingBlock` field working as designed.

Remarks

If `structSize` is less than the size of VOLUME_INFO, then only the first `structSize` bytes of VOLUME_INFO are returned.

The following equations explain how to calculate available disk space in bytes:

- Total usable blocks = `availableBlocks` + `purgableBlocks` .
- Block size in bytes = `sectorsPerBlock` * 512.
- TOTAL AVAILABLE DISK SPACE in bytes = total usable blocks * block size in bytes.

The VOLUME_INFO structure, pointed to by the `returnedVolumeStatistics` parameter, has the following format:

```
long   systemElapsedTime;
BYTE   volumeNumber;
BYTE   logicalDriveNumber;
WORD   sectorsPerBlock;
short  startingBlock;
LONG   totalBlocks;
LONG   availableBlocks;
LONG   totalDirectorySlots;
LONG   availableDirecotrySlots;
BYTE   isHashing;
BYTE   isRemovable;
BYTE   isMounted;
char   volumeName[17];
LONG   purgableBlocks;
LONG   notyetPurgableBlocks;
```

The *isRemovable* field always returns true.

See Also

GetVolumeInformation (page 57), GetVolumeInfoWithNumber (page 60), GetVolumeName (page 62), GetVolumeNumber (page 64)

Example

```
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>
#include <fcntl.h>
#include <nwshare.h>
#include <nwbitops.h>
#include <nwfile.h>
#include <nwdir.h>
#include <nwtts.h>
```

```
#include <nwbindry.h>
#include <time.h>

main()
 {
    int            rc;
    VOLUME_INFO    vs;
    char           svn[10];
    int            vn;

    printf("volume number: ");
    gets(svn);
    vn = atoi(svn);
    rc = GetVolumeStatistics(0,vn,sizeof (vs),&vs);
    if(rc)
    {
       printf("rc = %d\r\n",rc);
       printf("errno = %d\r\n",errno);
       printf("%s\r\n",strerror(errno));
    }
    else

   {
       printf("systemElapsedTime = %d\r\n",vs.systemElapsedTime);

       printf("volumeNumber = %d\r\n",vs.volumeNumber);

       printf("logicalDriveNumber = %d\r\n",vs.logicalDriveNumber);

       printf("sectorsPerBlock = %d\r\n",vs.sectorsPerBlock);

       printf("startingBlock = %d\r\n",vs.startingBlock);

       printf("totalBlocks = %d\r\n",vs.totalBlocks);
       printf("availableBlocks = %d\r\n",vs.availableBlocks);
       printf("totalDirectorySlots = %d\r\n",
              vs.totalDirectorySlots);
       printf("availableDirectorySlots = %d\r\n",
              vs.availableDirectorySlots);
       printf("isHashing = %d\r\n",vs.isHashing);
       printf("isRemovable = %d\r\n",vs.isRemovable);
       printf("isMounted = %d\r\n",vs.isMounted);

       printf("volumeName = %s\r\n",vs.volumeName);
    }
 }
```

NWGetExtendedVolumeInfo

Returns extended volume information

Local Servers: blocking

Remote Servers: blocking

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

Platform: NLM

Service: Volume

Syntax

```
#include <\nlm\nit\nwdir.h>

extern int NWGetExtendedVolumeInfo  (
   int                  connNumber,
   char                *volName,
   NWVolExtendedInfo   *volInfo);
```

Parameters

volNumber

(IN) Specifies the volume number.

volName

(IN) Specifies the volume name.

volInfo

(OUT) Points to NWVolExtendedInfo, which receives information.

Return Values

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

0x0000	SUCCESSFUL
0x8998	VOLUME_DOES_NOT_EXIST
0x897E	NCP_BOUNDARY_CHECK_FAILED
0x89FB	NO_SUCH_PROPERTY

Remarks

For more information, see NWVolExtendedInfo (page 45).

Revision History

A

The following table outlines all the changes that have been made to the Volume Management documentation (in reverse chronological order):

October 5, 2005	Transitioned to revised Novell documentation standards.
March 2, 2005	Fixed the legal information.
October 6, 2004	Fixed the preface.
February 18, 2004	Added links to sample code for the NWGetVolumeNumber (page 31) and NWGetExtendedVolumeInfo (page 19) functions.
October 8, 2003	Modified NWVolExtendedInfo (page 45) to indicate that the `statusFlag` field indicates whether the volume is an NSS volume.
July 30, 2003	Modified NWGetVolumeInfoWithNumber (page 26), NWGetVolumeInfoWithHandle (page 23), GetVolumeInformation (page 57), GetVolumeStatistics (page 66), and GetVolumeInfoWithNumber (page 60) to indicate that they do not return information about whether a volume is removable.
October 2002	Updated the information for NWScanMountedVolumeList (page 37), to clarify that this function is supported on NetWare 4.x and above.
September 2002	Updated the information for NWGetVolumeName (page 29) and the Pascal syntax for NWVolMountNumWithName (page 50).
May 2002	Updated the field descriptions of NWVolExtendedInfo (page 45).
February 2002	Updated links.
October 2001	Added Pascal syntax to NWScanMountedVolumeList (page 37) and NWVolMountNumWithName (page 50).
September 2001	Added support for NetWare 6.x to documentation.
June 2001	Updated tables.
February 2001	Added Chapter 5, “Server-Based Volume Management Functions,” on page 55, including volume functions moved from Multiple and Inter-File Services, and the server-based version of NWGetExtendedVolumeInfo (page 69). Added number of volumes allowed for NetWare 5.x (0-254) to NWGetVolumeInfoWithNumber (page 26), GetVolumeName (page 62), and GetVolumeNumber (page 64) along with an explanation. Replaced "bindery objects" references with "objects" since these references can also apply to NDS objects.
July 2000	Removed obsolete function NWGetVolumeStats from Chapter 3, “Functions,” on page 15.

May 2000	Added volume block size information and formula to NWGetExtendedVolumeInfo (page 19). Added restriction information to NWSetObjectVolSpaceLimit (page 40). Added Hi-Lo and byte swapping information to the `objectID` field of NWVOL_RESTRICTIONS (page 51). Fixed typographical errors in Return Values sections.
March 2000	Changed block sizes to 4K blocks and range to 0x40000000 in Remarks section of NWSetObjectVolSpaceLimit (page 40).
November 1999	Changed the description of NWGetObjDiskRestrictions (page 21) and added if the restriction is equal to 0x40000000 that the object has no restrictions.
June 1999	Added NWVolMountNumWithName (page 50) structure.