File Services APIs

Function Calls

This chapter describes the following File Services APIs. Those APIs which are not currently available on file servers running NetWare for UNIX software or NetWare v2.*x* are designated [386] for NetWare v3.*x*. Those that are not supported on file servers running NetWare v2.*x* but are supported on file servers running NetWare v3.*x* and NetWare for UNIX software are designated [386 & NWU].

Introduction to File Services

NetWare file services provide a set of supplementary calls that enable applications to manipulate files, directories, volumes, trustees and their associated information.

NetWare rights are checked before a client can perform any file service functions.

NWPath_t structure

The following structure is used to specify the location of NetWare file or directory:

```
typedef struct {
        NWDirHandle_ts      dirHandle;
        uint16              serverConnID;
        char                *pathName;
} NWPath_t;
```

dirHandle represents the directory handle allocated by the client pointing to a particular place in the directory structure.

serverConnID represents the file server which contains the file system being accessed.

pathName is a pointer which points to a character string which the client must allocate and fill in with a path name.

In order to specify a particular directory, the client can pass in any one of the following:

1) A dirHandle which points to the directory, and a zero-value path name (pointed to by the pathName field).

2) A dirHandle which points to a particular place in the directory structure, and then the path name (of sub-directories) beneath that place leading to the desired directory (pointed to by the pathName field).

3) A zero-value dirHandle and a full path name (pointed to by the pathName field).

Files are specified by adding the file's name to the path name (pointed to by the path name field).

NWClearObjectVolRestriction

This function clears any volume restrictions placed on an object with NWSetObjectVolRestriction. This function is supported in NetWare v3.*x* and above but is not currently supported in NetWare for UNIX software.

Synopsis

```
#include <nwapi.h>

int                     ccode;
```

```
uint16          serverConnID;
uint16          volNum;
uint32          objectID;
```

ccode=**NWClearObjectVolRestriction**(serverConnID, volNum, objectID);

Input

serverConnID passes the file server connection ID.

volNum passes the volume number.

objectID passes the object ID of the object whose restrictions you want to clear.

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Notes

The client must have security equivalence to SUPERVISOR.

See Also

NWGetVolNum
NWSetObjectVolRestriction

NWCloseFile

This function closes a file after it has been opened with NWOpenFile, NWCreateFile or NWCreateNewFile.

Synopsis

#include ªnwapi.hº

```
int             ccode;
uint16          serverConnID;
NWFileHandle_ta fileHandle;
```

ccode=**NWCloseFile**(serverConnID, fileHandle);

Input

serverConnID passes the file server connection ID.

fileHandle passes a pointer to the array containing the file handle.

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function closes a file after you have opened it with NWOpenFile, NWCreateFile or NWCreateNewFile and then deallocates the associated file handle. The file handle value is gained through either the NWCreateFile, NWCreateNewFile or the NWOpenFile function.

See Also

NWCreateFile
NWCreateNewFile
NWOpenFile

NWCreateDir

This function creates a NetWare directory on the server specified by the connection ID.

Synopsis

#include ªnwapi.hº

```
int             ccode;
NWPath_t        path;
uint16          inheritedRightsMask;

ccode=NWCreateDir( &path, inheritedRightsMask );
```

Input

path passes a pointer to the structure containing the directory handle, file server connection ID, and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

inheritedRightsMask passes the inherited rights mask for the new directory. (See Appendix A, ªTrustee Rights and Inherited Rights Mask.º)

Output

None.

Return Values

0 Successful.
-1 Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

The dirHandle parameter in NWPath_t Structure can be zero if the pathName parameter contains the complete path of the new directory, including the volume name. This call will not accept wildcard characters, and the requesting client must have the Create right in the directory that will become the parent directory.

This call will not sequentially create a string of directories; this call only creates the last directory provided in the NWPath_t structure provided by the client.

This call differs from NWCreateFile in that a handle is not returned. To obtain a directory handle to this directory, you must use NWAllocTemporaryDirHandle or NWAllocPermanentDirHandle.

See Also

NWDeleteDir

NWCreateFile

This function allows you to create a new file name and will overwrite an existing file of the same name.

Synopsis

#include ªnwapi.hº

```
int                 ccode;
NWPath_t            path;
uint32              fileAttributes;
NWFileHandle_ta     fileHandle;

ccode=NWCreateFile( &path, fileAttributes, fileHandle );
```

Input

path passes a pointer to the NWPath_t structure containing the file server connection ID, the directory handle and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

fileAttributes passes the file attributes of the file to be created. (See Appendix A, ªFile Attributes.º)

fileHandle passes a pointer to the array allocated for the fileHandle.

Output

fileHandle receives the file handle for the created file.

Return Values

0 Successful.

-1 Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function creates a new file by passing a file name and receiving a file handle. This function automatically opens a file allowing you to call NWWriteFile. This function will overwrite an existing file of the same name.

The function fails if the client does not have Create and Erase rights or if the file has the Delete Inhibit attribute set.

This function will not sequentially create a string of directories; this function will only create one file at a time. For example, in the path below, dir1 and dir2 must already have been created, or this call will fail:

volume:\dir1\dir2\filename

See Also

NWCreateNewFile

NWCreateNewFile

This function allows you to create a new file, but does not allow you to overwrite an existing file of the same name.

Synopsis

#include ªnwapi.hº

```
int                 ccode;
NWPath_t            path;
uint32              fileAttributes;
NWFileHandle_ta     fileHandle;

ccode=NWCreateNewFile( &path, fileAttributes, fileHandle );
```

Input

path passes a pointer to the NWPath_t structure containing the file server connection ID, the directory handle and the path name. (See Appendix A, ªNWPath_t Structure.º)

fileAttributes passes the file attributes of the file to be created. (See Appendix A, ªFile Attributes.º)

fileHandle passes a pointer to the array allocated for the fileHandle.

Output

fileHandle receives the file handle for the newly created file.

Return Values

0 Successful.
-1 Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	File Already Exists
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function creates a file name and file handle. This function also opens the file for writing with NWWriteFile.

This function fails if a file exists with the same name in the same directory or the client does not have the Create right in the parent directory. Use the NWCreateFile function if you want to overwrite files with the same name.

This function will not sequentially create a string of directories; this function will only create one file at a time. For example, in the path below, dir1 and dir2 must already have been created, or this call will fail:

volume:\dir1\dir2\filename

See Also

NWCloseFile
NWCreateFile

NWDeleteDir

This function deletes a NetWare directory.

Synopsis

#include ªnwapi.hº

```
int             ccode;
NWPath_t        path;

ccode=NWDeleteDir( &path );
```

Input

path passes a pointer to the NWPath_t structure containing the file server connection ID, the directory handle and the path name. (See Appendix A, ªNWPath_t Structure.º)

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function fails if any of the following conditions exist:

- The directory does not exist.
- Files exist in the existing directory.
- Another client has a directory handle pointing to the directory.
- The client does not have the Erase right to the target directory.
- The directory has the Delete Inhibit attribute set.

This function will not delete the volume root directory.

If the function succeeds, the function automatically deallocates any directory handles.

See Also

NWCreateDir

NWDeleteFile

This function marks a file for deletion.

Synopsis

#include ªnwapi.hº

```
int             ccode;
NWPath_t        path;
uint8           searchAttributes;

ccode=NWDeleteFile( &path, searchAttributes );
```

Input

path passes a pointer to the NWPath_t structure containing the file server connection ID, the directory handle, and a pointer to the path name. (See Appendix A, ªNWPath_t Structureº.)

searchAttributes passes the search attributes for the file, or files, to be deleted. (See Appendix A, ªSearch Attributes.º)

0x00	None (normal files)
0x02	Hidden
0x04	System
0x06	Both

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges

0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function deletes a NetWare file or group of files if wildcard characters are used by marking them for deletion and rendering them unviewable to the end user.

The searchAttributes parameter is used to include system and/or hidden files. In other words, if only the system bit is set in the searchAttributes parameter, then all files will be affected except hidden files. If only the hidden bit is set, all files will be affected except system files. When neither the hidden nor the system bit is set (0x00), then only files that are not hidden, system or both will be affected.

Notes

These files may be recovered with NWRecoverSalvageableFile unless one of the following conditions exist:

- The file server is running NetWare for UNIX software.
- The file server is low on disk space and the set time for saving a deleted file has passed. Under these conditions, the operating system will allow another client to overwrite the file.
- The Purge attribute has been set on the file(s) or the parent directory.
- The operating system has been configured to immediately purge all deleted files.

See Also

NWPurgeSalvageableFile
NWRecoverSalvageableFile

NWDeleteTrustee

This function removes a trustee from a directory's or file's trustee list.

Synopsis

#include ªnwapi.hº

```
int             ccode;
NWPath_t        path;
uint32          trusteeObjectID;

ccode=NWDeleteTrustee( &path, trusteeObjectID );
```

Input

path passes a pointer to the NWPath_t structure containing the file server connection ID, the directory handle and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

trusteeObjectID passes the trustee object ID of the trustee to be deleted.

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server
0xFE	No Trustee Exists

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function revokes a trustee's rights in a specific directory or file. The requesting client must have the Access Control right to the directory or file to delete a trustee. Deleting the explicit assignment of an object's trustee in a directory or file is not the same as assigning that object no rights in the directory. If no rights are assigned to a directory or file, the object inherits the rights it has in the parent directory minus the rights revoked with the directory's or file's Inherited Rights Mask.

Note

The trusteeObjectID can be obtained by calling NWGetObjectID.

See Also

NWGetEntrysTrustees
NWSetTrustee

NWFileCopy

This function copies a file, or portion of a file, to another file on the same file server.

Synopsis

#include ªnwapi.hº

```
int             ccode;
uint16          serverConnID;
NWFileHandle_ta sourceFileHandle;
NWFileHandle_ta destinationFileHandle;
uint32          sourceFileOffset;
uint32          destinationFileOffset;
uint32          numberOfBytesToCopy;
uint32          numberOfBytesCopied;

ccode=NWFileCopy( serverConnID, sourceFileHandle, destinationFileHandle,
sourceFileOffset, destinationFileOffset, numberOfBytesToCopy,
&numberOfBytesCopied);
```

Input

serverConnID passes the file server connection ID.

sourceFileHandle passes a pointer to the source file handle. (See ªDescriptionº on the next page.)

destinationFileHandle passes a pointer to the destination file handle. (See ªDescriptionº on the next page.)

sourceFileOffset passes the offset, in the source file, where the copying is to begin.

destinationFileOffset passes the offset, in the destination file, where the copying is to begin.

numberOfBytesToCopy passes the maximum number of bytes to copy.

numberOfBytesCopied passes a pointer to the space allocated for the number of bytes actually copied.

Output

numberOfBytesCopied returns a pointer to the number of bytes actually copied.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

The source and destination files must reside on the same file server. If they do not, the following error is returned:

NOT_SAME_CONNECTION

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

If the destination file is new, the numberOfBytesCopied parameter will return the size of the destination file; otherwise, the numberOfBytesCopied parameter will return the number of additional bytes added.

The sourceFileHandle should be obtained by calling NWOpenFile and passing NWOR_READ in the accessRights parameter. For this function to succeed, the client must have Read rights to the file.

The destinationFileHandle should be obtained by calling NWOpenFile and passing NWOR_WRITE in the accessRights parameter. For this function to succeed, the client must have Create rights in the parent directory.

Notes

This function only allows copying files on the same file server. If the client chooses to copy files across different file servers, the client must create the destination file (NWCreateFile), read from the source file (NWReadFile) and write to the destination file (NWWriteFile).

See Also

NWCloseFile
NWCreateFile
NWCreateNewFile
NWOpenFile
NWReadFile
NWWriteFile

NWGetDirEntryInfo

This function provides information about a directory through the directory handle. This function is supported in NetWare v3.*x* and in NetWare for UNIX software but not in NetWare v2.*x*.

Synopsis

#include ªnwapi.hº

```
int                 ccode;
uint16              serverConnID;
NWDirHandle_ts      dirHandle;
NWDirEntryInfo_t    dirInfo;

ccode=NWGetDirEntryInfo( serverConnID, dirHandle, &dirInfo );
```

Input

serverConnID passes the file server connection ID.

dirHandle passes the directory handle associated with the directory you are requesting information for.

dirInfo passes a pointer to the NWDirEntryInfo_t structure allocated for the directory entry information. (See Appendix A, ªNWDirEntryInfo_t Structure.º)

Output

dirInfo receives the directory entry information. (See Appendix A, ªNWDirEntryInfo_t Structure.º)

Return Values

0 Successful.

-1 Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Notes

This call is useful for obtaining information from the root directory.

The dirHandle parameter must be allocated using NWAllocTemporaryDirHandle or NWAllocPermanentDirHandle.

See Also

NWScanDirEntryInfo
NWAllocTemporaryDirHandle
NWAllocPermanentDirHandle

NWGetDirRestriction

This function checks for a directory's level and available blocks. This function is supported in NetWare v3.*x* but is not currently supported in NetWare for UNIX software.

Synopsis

#include ªnwapi.hº

```
int                 ccode;
uint16              serverConnID;
NWDirHandle_ts      dirHandle;
```

```
uint8                numberOfEntries;
NWDirRestriction_t   restrictions[n];
uint16               maxListElements;
```

ccode=**NWGetDirRestriction**(serverConnID, dirHandle, &numberOfEntries, restrictions, maxListElements);

Input

serverConnID passes the file server connection ID.

dirHandle passes the directory handle of the directory to be scanned.

numberOfEntries passes a pointer to the space allocated for the number of entries.

restrictions passes a pointer to the array of structures allocated for the directory restrictions. (See Appendix A, ªNWDirRestriction_t Structure.º)

maxListElements passes the maximum number of objects that you expect to have restrictions.

Output

numberOfEntries receives the number of entries actually copied into the restrictions parameter (0 - n).

restrictions receives the directory restrictions for each entry. (see Appendix A, ªNWDirRestriction_t Structure.º)

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function scans for the amount of disk space assigned to all directories between the current directory (referenced by the dirHandle) and the root directory. To find the actual amount of space available to a directory, scan all the entries returned in the restriction array and use the smallest one.

All directories will have a value in the maxBlocks parameter (from the NWDirRestriction_t structure). The maxBlocks parameter will return one of the following:

0x7FFFFFFF	No restrictions have ever been set.
negative value	Restrictions were set but they have been cleared. Use a zero in NWSetDirRestriction to clear restrictions.
positive value	Restrictions are set, and the positive value is the maximum value.

To calculate the amount of space in use, simply subtract availableBlocks from maxBlocks.

Notes

You must allocate a dirHandle before you make this call.

See Also

NWGetDirEntryInfo
NWScanDirEntryInfo
NWSetDirRestriction

NWGetEffectiveRights

This function returns the client's effective rights in the specified directory or file.

Synopsis

#include ªnwapi.hº

```
int             ccode;
NWPath_t        path;
uint16          effectiveRights;

ccode=NWGetEffectiveRights( &path, &effectiveRights );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, the file server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

effectiveRights passes a pointer to the space allocated for the effective rights for the directory or file. (See Appendix A, ªTrustee Rights and Inherited Rights Mask.º)

Output

effectiveRights receives the effective rights for the directory or file. (See Appendix A, ªTrustee Rights and Inherited Rights Mask.º)

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

For NetWare v3.*x*, the requesting workstation's effective rights are determined with the Inherited Rights Mask of the directory (or file), the client's trustee assignments and the trustee assignments of the groups the client belongs to.

- If the client has been granted a trustee assignment in a parent directory of the specified directory, the client's effective rights are the trustee rights of the parent directory minus any rights revoked by the specified directory's (or file's) inherited rights mask.
- If the client has been granted a trustee assignment to the specified directory (or file), the client's effective rights are the current trustee assignment.
- If the client belongs to a group, the group's effective rights are added to the client's effective rights.

Notes

For NetWare below v3.*x*, the effective rights to a file are always the same as the effective rights in the parent directory.

See Also

NWParseFullPath
NWDeleteTrustee
NWGetEntrysTrustees
NWSetTrustee

NWGetEntrysTrustees

This function scans an entry (directory or file) for trustees.

Synopsis

#include ªnwapi.hº

```
int                 ccode;
NWPath_t            path;
uint8               numberOfEntries;
NWTrusteeRights_t   trustees;
uint16              maxListElements;

ccode=NWGetEntrysTrustees( &path, &numberOfEntries, &trustees,
maxListElements );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, the server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

numberOfEntries passes a pointer to the space allocated for the number of entries found.

trustees passes a pointer to the space allocated for the trustees. (See Appendix A, ªNWTrusteeRights_t Structure.º)

maxListElements passes the maximum number of objects that you expect to have trustee rights.

Output

numberOfEntries receives the number of entries copied into the trusteeRights parameter (0 - n).

trusteeRights receives the trustee objectIDs and their associated rights. (See Appendix A, ªNWTrusteeRights_t Structure.º)

Return Values

0 Successful
-1 Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x9C	No Trustees
0xFE	Directory Locked
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function scans an entry for trustees and returns their objectID and trustee rights. For NetWare v3.*x*, this call may be made to directories or files, since trustees are assigned to files as well as directories. For versions below 3.*x*, this call may be made only to directories.

The client must have Access Control rights to the directory or file.

See Also

NWDeleteTrustee
NWSetTrustee

NWGetFileAttributes

This function returns a specified file's attributes.

Synopsis

#include ªnwapi.hº

```
int             ccode;
NWPath_t        path;
uint8           searchAttributes;
uint32          fileAttributes;

ccode=NWGetFileAttributes( &path, searchAttributes, &fileAttributes );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, the file server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

searchAttributes passes the search attributes of the file you are seeking. (See Appendix A, ªSearch Attributes.º)

0x00	None (normal files)
0x02	Hidden
0x04	System
0x06	Both

fileAttributes passes a pointer to the space allocated for the file's attributes. (See Appendix A, ªFile Attributes.º)

Output

fileAttributes receives the file's attributes. (See Appendix A, File Attributes.)

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible NetWare errors and a description of the four bytes in NWErrno.

Description

This function requires File Scan rights to the file.

See Also

NWSetFileAttributes

NWGetNameSpaceInfo

This function returns all name spaces and data stream information for the specified file server and volume. This function is supported in NetWare v3.*x*.

Synopsis

```
#include "nwapi.h"

int                 ccode;
uint16              serverConnId;
uint8               volNum;
NWNameSpaceInfo_t   nameSpace;

ccode=NWGetNameSpaceInfo( serverConnID, volNum, &nameSpace );
```

Input

serverConnID passes the file server connection ID.

volNum passes the number of the volume associated with the name space.

nameSpace passes a pointer to the structure allocated for the name space information. (See Appendix A, ªNWNameSpaceInfo_t Structure.º)

Output

nameSpace receives the name space information. (See Appendix A, ªNWNameSpaceInfo_t Structure.º)

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of thefour bytes in NWErrno.

Notes

The volNum can be obtained by calling NWGetVolNum.

See also

NWGetVolNum

NWGetObjectVolRestriction

This function gets the volume restrictions placed on a specified object (such as a user). This function is supported in v3.*x* but is not currently supported in NetWare for UNIX software.

Synopsis

```
#include "nwapi.h"

int         ccode;
uint16      serverConnID;
uint8       volNum;
uint32      objectID;
int32       restriction;
int32       inUse;

ccode=NWGetObjectVolRestriction( serverConnID, volNum, objectID, &restriction, &inUse );
```

Input

serverConnID passes the file server connection ID.

volNum passes the volume number.

objectID passes the object ID number for which the restrictions are being checked.

restriction passes a pointer to the space allocated for the object's volume restrictions.

inUse passes a pointer to the space allocated for the amount of volume space currently used by the object.

Output

restriction receives the object's volume restrictions on volume usage.

inUse receives the current amount of volume usage by the object.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function returns the amount of space restriction based on 4K blocks on a specified object as well as the current amount of space used by the object. If restriction value returned is equal to 0x40000000, there are no restrictions.

Clients can receive space restriction information about themselves, but a client must have security equivalence to SUPERVISOR to receive information about other objects.

Notes

The objectID can be obtained by calling NWGetObjectID.

The volNum can be obtained by calling NWGetVolNum.

See Also

NWGetVolNum
NWGetObjectID

NWGetVolInfoWithHandle

This function returns information about a volume based on a specified directory handle.

Synopsis

```
#include "nwapi.h"

int             ccode;
uint16          serverConnID;
NWDirHandle_ts  dirHandle;
NWVolUsage_t    volUsage;

ccode=NWGetVolInfoWithHandle( serverConnID, dirHandle, &volUsage );
```

Input

serverConnID passes the file server connection ID.

dirHandle passes the directory handle.

volUsage passes a pointer to the structure allocated for the volume usage information. (See Appendix A, ªNWVolUsage_t Structure.º)

Output

volUsage receives the volume usage information. (See Appendix A, ªNWVolUsage_t Structure.º)

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function returns information based on a directory handle. The information is placed in the NWVolUsage_t Structure.

Note: If the call is successful, the volume is mounted.

The following fields in the structure will always return a 0: purgableBlocks, notYetPurgableBlocks, maxDirEntriesUsed, volNum, isCached, isHashed, and isMounted.

Use NWGetVolNum to receive a valid value for the volume number.

For v3.*x*, use NWGetVolUsage to return valid values for the following fields: purgableBlocks and notYetPurgableBlocks.

For v2.*x* , use NWGetVolUsage to return valid values for the following fields: maxDirEntriesUsed, volNum, isCached, isHashed, and isMounted.

Notes

To obtain a directory handle, the application must call NWAllocTemporaryDirHandle or NWAllocPermanentDirHandle.

See Also

NWAllocPermanentDirHandle
NWAllocTemporaryDirHandle
NWGetVolNum
NWGetVolUsage

NWGetVolName

This function returns the name of the volume associated with the specified volume number.

Synopsis

```
#include "nwapi.h"

int             ccode;
uint16          serverConnID;
uint8           volNum;
char            volName[NWMAX_VOLUME_NAME_LENGTH];

ccode=NWGetVolName( serverConnID, volNum, volName );
```

Input

serverConnID passes the file server connection ID.

volNum passes the volume number for which the volume name is being obtained.

volName passes a pointer to the space allocated for the volume name (16 characters).

Output

volName receives the volume name.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle

0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function returns a volume's name by passing a volume number. The volNum parameter identifies the volume on the file server's Volume table. The Volume table contains information about each volume on the file server.

The volName parameter is 16 bytes long (including a null-byte). A volume name can be from 1 to 16 characters long and cannot include spaces or the following characters:

*****	(asterisk)
?	(question mark)
:	(colon)
/	(slash)
\	(backslash)

If a volume name is fewer than 16 characters long, the remaining characters in the volName parameter are null. If volName is 16 characters long, it is not null-terminated.

See Also

NWGetVolNum

NWGetVolNum

This function returns the volume number based on the file server connection ID number and the volume name. This call fails if the volume does not exist or the volume is not mounted.

Synopsis

```
#include "nwapi.h"

int                 ccode;
uint16              serverConnID;
char                volName[NWMAX_VOLUME_NAME_LENGTH];
uint8               volNum;

ccode=NWGetVolNum( serverConnID, volName, &volNum );
```

Input

serverConnID passes the file server connection ID.

volName passes a pointer to the volume name. Do not include a colon with the volume name.

volNum passes a pointer to the space allocated for the volume number.

Output

volNum receives the volume number.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x98	Volume Does Not Exist
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function returns a volume's number based on the file server serverConnID and volume name.

The volume name cannot contain wildcards.

If the volNum parameter is between 0 and the maximum allowable volume number on the network, the call is successful and a zero is returned.

See Also

NWGetVolName

NWGetVolsObjectRestrictions

This function will scan a volume for any object restrictions. This function is supported in v3.*x* but is not currently supported in NetWare for UNIX software.

Synopsis

#include "nwapi.h"

```
int                 ccode;
uint16              serverConnID;
uint8               volNum;
uint8               numberOfEntries;
NWUserRestriction_t restrictions[n];
uint16              maxListElements;

ccode=NWGetVolsObjectRestrictions( serverConnID, volNum,
&numberOfEntries, restrictions, maxListElements );
```

Input

serverConnID passes the file server connection ID.

volNum passes the volume number.

numberOfEntries passes a pointer to the space allocated for the number of entries.

restrictions passes a pointer to the array of structures allocated for the user restrictions. (See Appendix A, ªNWUserRestriction_t Structure.º)

maxListElements asses the maximum number of objects that you expect to have restrictions.

Output

numberOfEntries receives the number of entries that were copied into the restrictions array (0 - n).

restrictions receives the user restrictions. (See Appendix A,

ªNWUserRestriction_t Structure.º)

Return Values

0 Successful.
-1 Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function returns a list of the object restrictions for a specified volume. All restrictions are in 4K blocks. A restriction may be zero.

Notes

The volNum can be obtained by calling NWGetVolNum.

The client must have security equivalence to SUPERVISOR.

See Also

NWGetVolNum
NWSetObjectVolRestriction

NWGetVolUsage

This function gives you information about what is available, and in use, on a certain volume.

Synopsis

```
#include "nwapi.h"

int             ccode;
uint16          serverConnID;
uint8           volNum;
NWVolUsage_t    volUsage;

ccode=NWGetVolUsage( serverConnID, volNum, &volUsage );
```

Input

serverConnID passes the file server connection ID.

volNum passes the volume number of the volume being checked.

volUsage passes a pointer to the structure allocated for the volume usage information. (See Appendix A, ªNWVolUsage_t Structure.º)

Output

volUsage receives the filled-in structure with the volume usage information. (See Appendix A, ªNWVolUsage_t Structure.º)

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x98	Volume Does Not Exist (NetWare v3.*x* and NWU)
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

The volNum parameter identifies the volume on the file server's volume table, which contains information about each volume on the file server. Use NWGetVolNum to obtain a volume number.

For NetWare for UNIX software, the following fields in the NWVolUsage_t Structure will always return a 0: purgableBlocks, notYetPurgableBlocks, maxDirEntriesUsed, sectorsPerBlock, isHashed, isCached, isRemovable, isMounted.

For NetWare v3.*x*, the following fields in the NWVolUsage_t Structure will always return a 0: maxDirEntriesUsed, isCached, isHashed, isRemovable, isMounted. Use NWGetVolInfoWithHandle to return a valid value for isRemovable. This call fails if the volume does not exist or the volume isn't mounted.

For NetWare v2.*x*, the following fields in the NWVolUsage_t Structure will always return a 0: purgableBlocks, notYetPurgableBlocks, and sectorsPerBlock. Use NWGetVolInfoWithHandle to return a valid value for sectorsPerBlock.

See Also

NWGetVolNum
NWGetVolInfoWithHandle

NWMoveEntry

This function allows you to move and rename a file or directory. This function is supported in NetWare v3.*x* and in NetWare for UNIX software but not in NetWare v2.*x*.

Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;
uint8           searchAttributes;
NWDirHandle_ts  newDirHandle;
char            newPathName[NWMAX_DIR_PATH_LENGTH];

ccode=NWMoveEntry( &path, searchAttributes, newDirHandle,
newPathName );
```

Input

path passes a pointer to the structure containing the directory handle, server connection ID and a pointer to the path name of the source file or directory. (See Appendix A, ªNWPath_t Structure.º)

search Attributes passes the search attributes for hidden or system files or directories. (See "Description" on the next page and Appendix A, ªSearch Attributes.º)

none	0x00
hidden	0x02
system	0x04
both	0x06
directories	0x10
files	0x20

newDirHandle passes the new directory handle of the destination file or directory. (See "Description" on the next page.)

newPathName passes a pointer to the destination file or directory name (See "Description" on the next page.)

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

The path parameter specifies the source file or directory that the client wants to move. The client must have Erase rights to the source directory or file and Create rights in the destination directory.

The searchAttributes parameter must contain either the directory's or file's search attribute bit so that the file server knows whether a file or directory is being moved. The directory's or file's search attribute bit may be OR'ed with the other search attribute values if you want to move system or hidden files and directories.

The newDirHandle parameter is the directory handle for the destination of the file or directory. The destination directory must be on the same file server and volume as the source directory. The newDirHandle parameter can contain a zero value if a full path is passed in the newPathName parameter. To use a value other than zero, the newDirHandle must be allocated using NWAllocPermanentDirHandle or NWAllocTemporaryDirHandle.

The newPathName parameter is the new name for the directory or file in its new destination. If zero is passed in the newDirHandle parameter, a full path can be specified in the newPathName parameter. If a value other than zero is passed in the newDirHandle parameter, the newPathName parameter should specify only the directory or file name.

See Also

NWAllocPermanentDirHandle
NWAllocTemporaryDirHandle
NWMoveFile
NWRenameDir

NWMoveFile

This function moves or renames a file.

Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;
uint8           searchAttributes;
NWDirHandle_ts  destDirHandle;
char            destFileName[NWMAX_FILE_NAME_LENGTH];

ccode=NWMoveFile( &path, searchAttributes, destDirHandle, destFileName );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name of the source file. (See Appendix A, ªNWPath_t Structure.º)

searchAttributes passes the search attributes for hidden or system files. (See Appendix A, ªSearch Attributes.º)

none	0x00
hidden	0x02
system	0x04
both	0x06

destDirHandle passes the directory handle of the destination directory. (See "Description" on the next page.)

destFileName passes a pointer to the destination file name. (See "Description" on the next page.)

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

You can use this function to rename a file by simply moving it to the same directory with a new file name. If you

use this call to move a file, the destination directory must reside on the same server and volume as the source directory.

The destDirHandle may contain a zero value if a full path is passed in the destFileName parameter. To obtain a directory handle, call NWAllocTemporaryDirHandle or NWAllocPermanentDirHandle.

The searchAttributes parameter is used to include system and/or hidden files. In other words, if only the system bit is set in the searchAttributes parameter all files will be affected except hidden files. If only the hidden bit is set, all files will be affected except system files. When neither the hidden nor the system bit is set (0x00), then only files that are not hidden, system or both will be affected.

Notes

To move a file to a different server, the application must create a file on the target server (NWCreateFile or NWCreateNewFile) and then read from the source file (NWReadFile) and write to the destination file (NWWriteFile).

See also

NWAllocTemporaryDirHandle
NWAllocPermanentDirHandle
NWCreateFile
NWCreateNewFile
NWReadFile
NWWriteFile

NWOpenFile

This function opens a previously created file for reading or writing.

Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;
uint8           searchAttributes;
uint8           accessRights;
NWFileHandle_ta fileHandle;

ccode=NWOpenFile( &path, searchAttributes, accessRights, fileHandle );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

searchAttributes passes the search attributes for hidden or system files. (See Appendix A, ªSearch Attributes.º)

none	0x00
hidden	0x02
system	0x04
both	0x06

accessRights passes the access rights of the file to open. (See Appendix A, ªOpen Access Rights.º)

fileHandle passes a pointer to the space allocated for the file handle of the file to be opened.

Output

fileHandle receives the file handle of the file to be opened.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x94	Invalid Open Access Rights
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function opens a file and returns the file handle for reading or writing.

Notes

You should use NWCloseFile after completing the read or write to the file.

See Also

NWCloseFile
NWReadFile
NWWriteFile

NWPurgeSalvageableFile

This function permanently deletes files that have been erased but are still recoverable. This function is supported in NetWare v3.*x* but is not currently supported in NetWare for UNIX software.

Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;
int32           entryID;

ccode=NWPurgeSalvageableFile( &path, entryID );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID, and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

entryID passes the entryID for the file to be purged.

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

When a file is deleted, it can still be recovered for a time period and still uses disk space. This function can be used

- To permanently delete erased but recoverable files.
- To free the disk space being used by deleted, but still recoverable, files.

This function purges one file previously marked for deletion. Use NWScanSalvageableFiles until you find a file you want to purge. Then call NWPurgeSalvageableFile, passing in the entryID that corresponds to the desired file. The entryID is obtained from NWScanSalvageableFiles.

See Also

NWRecoverSalvageableFile
NWScanSalvageableFiles

NWReadFile

This function allows you to read a file.

Synopsis

```
#include "nwapi.h"

int             ccode;
uint16          serverConnID;
NWFileHandle_ta fileHandle;
uint32          startingOffset;
uint32          bytesToRead;
uint32          bytesActuallyRead;
char            data[n];

ccode=NWReadFile( serverConnID, fileHandle, &startingOffset, bytesToRead, &bytesActuallyRead, data );
```

Input

serverConnID passes the file server connection ID.

fileHandle passes a pointer to the file handle.

startingOffset passes the address of the offset where the read should begin.

bytesToRead passes the maximum number of bytes to be read (should not exceed n).

bytesActuallyRead passes a pointer to the space allocated for the actual number of bytes read.

data passes a pointer to the space allocated for the data being read.

Output

startingOffset receives the new starting offset (the previous offset plus the bytes that were read).

bytesActuallyRead receives the actual number of bytes that were read (0 - n).

data receives the data that is read.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function reads from a file that has been previously opened with NWCreateFile, NWCreateNewFile or NWOpenFile.

See Also

NWCloseFile
NWCreateFile
NWCreateNewFile
NWOpenFile
NWWriteFile

NWRecoverSalvageableFile

This function restores a deleted but salvageable file. This function is supported in 3.*x* but is not currently supported in NetWare for UNIX software.

Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;
int32           entryID;
char            newFileName[NWMAX_FILE_NAME_LENGTH];

ccode=NWRecoverSalvageableFile( &path, entryID, newFileName );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID, and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

entryID passes the entryID corresponding to the file. (See "Description" below.)

newFileName passes a pointer to the space allocated for the file name. This space contains the name of the file to be restored. (This name may be the same name as the salvageable file's, unless another file was created with the same name.)

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function restores one file previously marked for deletion. You should use NWScanSalvageableFiles until you find a file you want to salvage. Then call NWRecoverSalvageableFile, passing in the entryID that corresponds to the desired file. The entryID is obtained from NWScanSalvageableFiles.

Notes

If the client creates more than one file with the same name as an erased file, the function renames the erased files, replacing the last two characters of the file extension with 00. For example,

TEST.DAT	becomes	TEST.D00
TEST	becomes	TEST.00

See Also

NWPurgeSalvageableFile
NWScanSalvageableFiles

NWRenameDir

This function allows you to change the name of a directory.

Synopsis

```
#include "nwapi.h"

int         ccode;
NWPath_t    path;
char        newDirName[NWMAX_DIR_NAME_LENGTH];

ccode=NWRenameDir( &path, newDirName );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

newDirName passes a pointer to the array allocated for the new directory name.

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

The newDirName parameter should only contain the directory's new name, not a path specification. Names longer than the DOS 8.3 character naming convention will be truncated.

This function will rename, not move, a directory. To move a directory, see NWMoveEntry.

See Also

NWMoveEntry

NWScanDirEntryInfo

This function scans for directory entry information such as entry names, attributes, creation (date and time), archive (date and time) and last modification (date and time).

Synopsis

```
#include "nwapi.h"

NWBoolean_ts        ccode;
NWPath_t            path;
int32               entryID;
uint8               searchAttributes;
NWDirEntryInfo_t    dirInfo;

entryID=-1;

ccode=NWScanDirEntryInfo( &path, &entryID, searchAttributes, &dirInfo )
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name. (See "Description" on the next page and Appendix A, ªNWPath_t Structure.º)

entryID passes a pointer to the entryID of the previously scanned directory.(See "Description" on the next page.)

searchAttributes passes the search attributes. (See Appendix A, ªSearch Attributes.º)

none	0x00
hidden	0x02
system	0x04
both	0x06

dirInfo passes a pointer to the structure allocated for the directory entry information. (See Appendix A, ªNWDirEntryInfo_t Structure.º)

Output

entryID receives the entryID of the current directory.

dirInfo receives the directory entry information. (See Appendix A, ªNWDirEntryInfo_t Structure.º)

Return Values

1 Successful
0 Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

The application must provide a search string in the pathName field of the NWPath_t structure. Use the following examples to determine the search string you want to specify in the pathName field:

Scan all directories in the sys volume	sys:*
Scan all directories under sys:dir1:	sys:\dir1*
Scan directories under dir1 beginning with t	sys:\dir1\t*
Scan information on dir2:	sys:\dir1\dir2

The entryID parameter should pass in a -1 for the first scan; for subsequent calls, the entryID of the previously scanned directory should be passed. This entryID only has meaning for the file server. The application should not have to manipulate this value.

Notes

For applications talking to v2.*x* servers, the following fields in the NWDirEntryInfo_t structure will contain valid values:

entryName
creationDateAndTime
ownerID
inheritedRightsMask

The remaining structure members will be zero-filled.

See Also

NWGetDirEntryInfo
NWSetDirEntryInfo

NWScanFileEntryInfo

This function returns information about a file such as owner, size, attributes, last access (date and time) and creation

(date and time).

Synopsis

```
#include "nwapi.h"

NWBoolean_ts        ccode;
NWPath_t            path;
int32               entryID;
uint8               searchAttributes;
NWFileEntryInfo_t   fileEntryInfo;

entryID=-1;

ccode=NWScanFileEntryInfo( &path, &entryID, searchAttributes,
&fileEntryInfo );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name. (See "Description" on the next page and Appendix A, ªNWPath_t Structure.º)

entryID passes a pointer to the entryID of the previously scanned file. (See "Description" on the next page.)

searchAttributes passes the search attributes. (See Appendix A, ªSearch Attributes.º)

none	0x00
hidden	0x02
system	0x04
both	0x06

fileEntryInfo passes a pointer to the structure allocated for the file entry information. (See Appendix A, ªNWFileEntryInfo_t Structure.º)

Output

entryID receives the sequence number of the current file.

fileEntryInfo receives the file entry information. (See Appendix A, ªNWFileEntryInfo_t Structure.º)

Return Values

1 Successful.

0 Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

The application must provide a search string in the pathName field of the NWPath_t structure. Use the following examples to determine the search string you want to specify in the pathName field:

Scan all files in the sys:dir1 directory	sys:dir1*
Scan all files in the sys:dir1 directory beginning with t	sys:dir1\t*

The entryID parameter should pass in a -1 for the first scan; for subsequent calls, the entryID of the previously scanned file should be passed. This entryID only has meaning for the file server. The application should not have to manipulate this value.

Notes

The fileSize field in the NWFileEntryInfo_t structure contains the logical file size.

For applications talking to NetWare v 2.*x* file servers, the following fields in the NWFileEntryInfo_t structure will be zero:

archiverID
updatorID
inheritedRightsMask
nameSpaceID

See Also

NWSetFileEntryInfo

NWScanSalvageableFiles

This function returns information on deleted but salvageable files. This function is supported in v3.*x* but is not currently supported in NetWare for UNIX software.

Synopsis

```
#include "nwapi.h"

NWBoolean_ts        ccode;
uint16              serverConnID;
NWDirHandle_ts      dirHandle;
int32               entryID;
NWSalvageableInfo_t salvageInfo;

entryID=-1;

ccode=NWScanSalvageableFiles( serverConnID, dirHandle, &entryID,
&salvageInfo );
```

Input

serverConnID passes the file server connection ID.

dirHandle passes the directory handle of the directory to be scanned.

entryID passes the entryID of the previously scanned salvageable file. (See "Description" on the next page.)

salvageInfo passes a pointer to the structure allocated for the salvageable entry information. (See Appendix A, ªNWSalvageableInfo_t Structure.º)

Output

entryID receives the entryID of the current salvageable file. (See "Description" on the next page.)

salvageInfo receives the structure allocated for the salvageable entry information. (See Appendix A, ªNWSalvageableInfo_t Structure.º)

Return Values

1 Successful.
0 Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

The entryID parameter should pass in a -1 for the first scan; for subsequent calls, the entryID of the previously scanned file should be passed. This entryID only has meaning for the file server. The application should not have to manipulate this value.

See Also

NWRecoverSalvageableFile
NWPurgeSalvageableFile

NWScanTrusteePaths

This function returns the directory paths to which an object has trustee rights.

Synopsis

```
#include "nwapi.h"

NWBoolean_ts    ccode;
uint16          serverConnID;
uint32          objectID;
uint8           volNum;
int32           entryID;
uint16          trusteeAccessRights;
char            directoryPath[NWMAX_DIR_PATH_LENGTH];

entryID=-1;

ccode=NWScanTrusteePaths( serverConnID, objectID, volNum, &entryID, &trusteeAccessRights,
directoryPath );
```

Input

serverConnID passes the server connection ID.

objectID passes the object ID of the user or group for which the trustee information is to be found.

volNum passes the volume number of the volume being searched.

entryID passes a pointer to the entryID of the previously scanned directory path. (See "Description" on the next page.)

trusteeAccessRights passes a pointer to the space allocated for the trustee's access mask. (See Appendix A, ªTrustee Rights and Inherited Rights Mask.º)

directoryPath passes a pointer to the space allocated for the current trustee's directory path name.

Output

entryID receives the entryID of the current directory. (See "Description" on the next page.)

trusteeAccessRights receives the trustee's access mask. (See Appendix A, ªTrustee Rights and Inherited Rights Mask.º)

directoryPath receives the current trustee's directory path name.

Return Values

1 Successful.
0 Unsuccessful. One of the following error codes is placed in NWErrno:

0xFC	No Such Object
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function is used iteratively to determine all of a bindery object's trustee directory paths and corresponding access masks.

The entryID parameter should pass in a -1 for the first scan; for subsequent calls, the entryID of the previously scanned directory path should be passed. This entryID only has meaning for the file server. The application should not have to manipulate this value.

Notes

Only SUPERVISOR, the object, or a bindery object that is security equivalent to SUPERVISOR or the object can scan an object's trustee directory paths.

The objectID can be obtained by calling NWGetObjectID.

The volNum can be obtained by using NWGetVolNum.

See Also

NWGetObjectID
NWGetVolNum

NWSetDirEntryInfo

This function sets or changes information kept about a directory such as owner, attributes, creation (date and time) or last access (date and time).

Synopsis

```
#include "nwapi.h"

int                     ccode;
NWPath_t                path;
```

```
uint8                searchAttributes;
uint32               changeAttributes;
NWDirEntryInfo_t     dirEntryInfo;

ccode=NWSetDirEntryInfo( &path, searchAttributes, changeAttributes, &dirEntryInfo);
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID, and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

searchAttributes passes the search attributes for any hidden or system files. (See Appendix A, ªSearch Attributes.º)

none	0x00
hidden	0x02
system	0x04
both	0x06

changeAttributes passes the new attributes for the directory entry. (See Appendix A, ªChange Attributes.º)

dirEntryInfo passes a pointer to the structure allocated for the directory entry information. (See Appendix A, ªNWDirEntryInfo_t Structure.º)

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function sets information kept on directories. This information can be seen in the NWDirEntryInfo_t structure. A pointer to the entire structure should be passed, even if only one item is being changed. Furthermore, the change attributes must be passed which correspond to the data being changed. For example, if the owner of the directory is being changed, the entire structure would be allocated and the new ownerID would be put in the ownerID field of the structure. Correspondingly, the NWCA_OWNER_ID change attribute would be passed in the changeAttributes parameter. (The ownerID is the object ID of the owner.)

If you use this function to change the name of a directory, you must change the path name to the new directory name for subsequent calls.

Notes

If more than one item is being changed, the change attributes may be OR'ed together. If you only want to change the directory's inherited rights mask, use NWSetDirsInheritedRightsMask.

For v2.*x* servers, the only data which can be changed (as referenced in the NWDirEntryInfo_t structure) is:

creationDateAndTime
ownerID
inheritedRightsMask

For v3.*x* servers, the data that cannot be changed is:

nameSpaceID

See Also

NWScanDirEntryInfo
NWSetDirsInheritedRightsMask

NWSetDirRestriction

This function sets (or clears) a directory's restrictions. This function is supported in v3.*x* but is not currently supported in NetWare for UNIX software.

Synopsis

```
#include "nwapi.h"

int             ccode;
uint16          serverConnID;
NWDirHandle_ts  dirHandle;
int32           restriction;

ccode=NWSetDirRestriction( serverConnID, dirHandle, restriction );
```

Input

serverConnID passes the file server connection ID.

dirHandle passes the directory handle of the directory that will have its restrictions set.

restrictions passes the restrictions.

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function requires that the application pass an allocated directory handle to the directory to which the restrictions apply. The directory handle can be obtained by calling either NWAllocTemporaryDirHandle or NWAllocPermanentDirHandle.

The restriction parameter passes a 0 to clear all restrictions or a number corresponding to the space available in the directory. Restrictions are in 4K blocks; therefore, a restriction of 1 will restrict the space usage in a particular directory to 4K.

See Also

NWGetDirRestriction
NWAllocTemporaryDirHandle
NWAllocPermanentDirHandle

NWSetDirsInheritedRightsMask

This function sets the rights mask for a directory path.

Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;
uint16          newRightsMask;

ccode=NWSetDirsInheritedRightsMask( &path, newRightsMask );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

newRightsMask passes the rights that you want to grant the directory's rights mask. (See Appendix A, ªTrustee Rights and Inherited Rights Mask.º)

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function modifies an Inherited Rights Mask for a directory by replacing the existing mask. The function does not add or subtract rights from the existing mask. You pass all rights you want in the rights mask with the newRightsMask parameter.

See Also

NWGetDirEntryInfo
NWScanDirEntryInfo
NWSetFilesInheritedRightsMask

NWSetFileAttributes

This function modifies a file's attributes.

Synopsis

#include "nwapi.h"

```
int             ccode;
NWPath_t        path;
uint8           searchAttributes;
uint32          fileAttributes;

ccode=NWSetFileAttributes( &path, searchAttributes, fileAttributes );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

searchAttributes passes the search attributes. (See Appendix A, ªSearch Attributes.º)

none	0x00
hidden	0x02
system	0x04
both	0x06

fileAttributes passes the file attributes to be set on the file designated in the pathName field in the NWPath_t structure. (See Appendix A, ªFile Attributes.º)

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

The searchAttributes parameter is used to include system and/or hidden files. In other words, if only the system bit is set in the searchAttributes parameter all files will be affected except hidden files. If only the hidden bit is set, all files will be affected except system files. When neither the hidden nor the system bit is set (0x00), only files that are not hidden, system or both will be affected.

See Also

NWSetFileEntryInfo
NWScanFileEntryInfo

NWSetFileEntryInfo

This function sets file information such as owner, creation (date and time) and last access (date and time).

Synopsis

#include "nwapi.h"

```
int                 ccode;
NWPath_t            path;
uint8               searchAttributes;
uint32              changeAttributes;
NWFileEntryInfo_t   fileInfo;

ccode=NWSetFileEntryInfo( &path, searchAttributes, changeAttributes, &fileInfo );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

searchAttributes passes the search attributes. (See Appendix A, ªSearch Attributes.º)

none	0x00
hidden	0x02
system	0x04
both	0x06

changeAttributes passes the change attributes. (See "Description" on the next page and Appendix A, ªChange Attributes.º)

fileInfo passes a pointer to the structure allocated for the file informationbeing set. (See Appendix A, ªNWFileEntryInfo_t Structure.º)

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function sets information kept on files. This information can be seen in the NWFileEntryInfo_t structure. A pointer to the entire structure should be passed, even if only one item is being changed. Furthermore, the change

attributes must be passed which correspond to the data being changed. For example, if the owner of the file is being changed, the entire structure would be allocated and the new ownerID would be put in the ownerID field of the structure. Correspondingly, the NWCA_OWNER_ID change attribute would be passed in the changeAttributes parameter.

Notes

To only change the file attributes, use NWSetFileAttributes. To only change the file's Inherited Rights Mask, use NWSetFilesInheritedRightsMask.

For all versions of servers, the following data cannot be changed:

fileSize
nameSpaceID

In addition, for v2.*x* , the following data cannot be changed:

entryName
archiverID
updatorID
updateDateAndTime
inheritedRightsMask

See Also

NWScanFileEntryInfo
NWSetFileAttributes
NWSetFilesInheritedRightsMask

NWSetFilesInheritedRightsMask

This function sets the rights mask for a file. This function is supported in NetWare v3.*x*.

Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;
uint16          newRightsMask;

ccode=NWSetFilesInheritedRightsMask( &path, newRightsMask );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

newRightsMask passes the rights that you want to grant the file's rights mask. (See Appendix A, ªTrustee Rights and Inherited Rights Mask.º)

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function modifies an Inherited Rights Mask for a file by replacing the existing mask. The function does not add or subtract rights from the existing mask. You pass all rights you want in the rights mask with the newRightsMask parameter.

Notes

This function is only valid for v3.*x* and NetWare for UNIX software. On versions below 3.*x*, inherited rights cannot be set on files.

See Also

NWScanFileEntryInfo
NWSetFileAttributes
NWSetFileEntryInfo

NWSetObjectVolRestriction

This function sets restrictions on objects in a volume. This function is supported in v3.*x* but is not currently supported in NetWare for UNIX software.

Synopsis

```
#include "nwapi.h"

int         ccode;
uint16      serverConnID;
uint8       volNum;
uint32      objectID;
int32       restriction;

ccode=NWSetObjectVolRestriction( serverConnID, volNum, objectID,
restriction );
```

Input

serverConnID passes the file server connection ID.

volNum passes the volume number.

objectID passes the bindery object ID of the object for which the restrictions are being set.

restriction passes the objects volume restrictions.

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function is similar to NWSetDirRestriction in that a space restriction is set, but the restriction applies to a specific object rather than overall. Restrictions are set in 4K blocks.

Notes

The objectID can be obtained by calling NWGetObjectID.

The volNum can be obtained by calling NWGetVolNum

The client must have security equivalence to SUPERVISOR.

See Also

NWGetVolNum
NWGetObjectID
NWClearObjectVolRestriction
NWGetObjectVolRestriction

NWSetTrustee

This function creates a trustee for a file or directory. It also can change a current trustee's trustee rights.

Synopsis

```
#include "nwapi.h"

int             ccode;
NWPath_t        path;
uint32          trusteeObjectID;
uint16          trusteeRightsMask;

ccode=NWSetTrustee( &path, trusteeObjectID, trusteeRightsMask );
```

Input

path passes a pointer to the NWPath_t structure containing the directory handle, server connection ID and a pointer to the path name. (See Appendix A, ªNWPath_t Structure.º)

trusteeObjectID passes the bindery object ID of the trustee.

trusteeRightsMask passes the trustee rights mask. (See "Description" on the next page and Appendix A, ªTrustee Rights and Inherited Rights Mask.º)

Output

None.

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function assigns a user with specific rights to a directory or file; the user may already be a trustee in the directory or may be a new trustee. Once assigned, the user is called a trustee of that directory. To take rights away from a trustee, simply pass a trusteeRightsMask without those trustee rights bits set.

If the trustee (represented by the trusteeObjectID) is already a trustee for this directory, all current trustee assignments will be replaced with the new trusteeRightsMask. This call will not add the new trusteeRightsMask to the current trustee rights.

Notes

The trusteeRightsMask is a value that can be obtained by ORing together all of the desired trustee rights. (See Appendix A, ªTrustee Rights and Inherited Rights Maskº.)

The trusteeObjectID can be obtained using NWGetObjectID.

See Also

NWGetObjectID

NWWriteFile

This function allows you to write to a file.

Synopsis

```
#include "nwapi.h"

int                 ccode;
uint16              serverConnID;
NWFileHandle_ta      fileHandle;
uint32              startingOffset;
uint32              bytesToWrite;
char                data[n];

ccode=NWWriteFile( serverConnID, fileHandle, &startingOffset,
bytesToWrite, data );
```

Input

serverConnID passes the file server connection ID.

fileHandle passes a pointer to the file handle.

startingOffset passes a pointer to the offset where the write is supposed to begin.

bytesToWrite passes the number of bytes to write.

data passes a pointer to data being written.

Output

startingOffset receives the new starting offset (the previous offset plus the bytes that were written).

Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:

0x9C	Invalid Path
0xFF	No Files Found
0x84	No Create Privileges
0x9B	Bad Directory Handle
0x9E	Invalid Filename
0xF8	Not Attached To Server

See Appendix B for a complete listing of possible errors and a description of the four bytes in NWErrno.

Description

This function will write to a file after the file has been created and opened.

Notes

You must first use NWCreateFile, NWCreateNewFile or NWOpenFile to get a file handle for the file to be written to. The file should be closed using
NWCloseFile after it has been written to.

See Also

NWCloseFile
NWCreateFile
NWCreateNewFile
NWReadFile