

# Path Services APIs

## Introduction to Path Services

Path Services system calls provide developers with the necessary tools to do the following:

- Allocate directory handles
- Deallocate directory handles
- Return information about directory paths

Path Services system calls operate through a series of tables maintained by both the file server and the client. Clients must maintain tables which keep track of the directory handles they create. Directory handles represent a full directory path name and can be used as a convenient method for pointing to a particular place level in the directory structure. Most calls in the File System Services allow you to use directory handles when specifying a file or directory. The calls usually accept the directory handle in the `NWPath_t` structure.

### 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).

## NWAllocPermanentDirHandle

This function assigns a permanent directory handle.

### Synopsis

```
#include "nwapi.h"

int              ccode;
NWPath_t        path;
```

```

uint16      driveNum;
NWDirHandle_ts  newDirHandle;
uint16      effectiveRightsMask;

```

```

ccode=NWAllocPermanentDirHandle( &path, driveNum, &newDirHandle,  &effectiveRightsMask );

```

## Input

*path* passes a pointer to the NWPath\_t structure created for the directory handle, the server connection ID, and a pointer to the directory path.

*driveNum* passes the drive number.

*newDirHandle* passes a pointer to the structure allocated for the new directory handle.

*effectiveRightsMask* passes a pointer to the space allocated for the client's effective rights to the directory connected via the newDirHandle parameter. See "Trustee Rights and Inherited Rights Mask" in Appendix A.

## Output

*newDirHandle* receives the new directory handle.

*effectiveRightsMask* receives the client's effective rights to the directory connected via the newDirHandle parameter. See "Trustee Rights and Inherited Rights Mask" in Appendix A.

## Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:
0x98	Volume Does Not Exist
0x9C	Invalid Path

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

## Description

The directory handles allocated by this call are permanently allocated to the client's object ID by the file server unless deallocated by a call to

NWDeallocateDirHandle, or overwritten by allocating over the same dirHandle. The client must keep track of dirHandles and the directories they represent for use with other function calls.

The driveNum parameter should be a unique number between 1-32. This number is required by the server when allocating dirHandles. The number could be used by the client in a table to represent a particular dirHandle, although the actual dirHandle must always be used when sending requests to the server.

The effectiveRightsMask receives the client's effective rights to the directory as determined by ANDing the directory's Inherited Rights Mask and the client's trustee rights in that directory.

## Notes

If a volume is specified in the path name, the volume must be represented by the volume name followed by a colon. You should pass a different drive number for each dirHandle allocated to avoid overwriting an existing dirHandle.

## See Also

NWAllocTemporaryDirHandle  
 NWDeallocateDirHandle  
 NWParseFullPath

## NWAllocTemporaryDirHandle

This function assigns a temporary directory handle.

### Synopsis

```
#include "nwapi.h"

int          ccode;
NWPath_t     path;
uint16       driveNum;
NWDirHandle_ts newDirHandle;
uint16       effectiveRightsMask;

ccode=NWAllocTemporaryDirHandle( &path, driveNum, &newDirHandle, &effectiveRightsMask );
```

### Input

*path* passes a pointer to the NWPath\_t structure created for the directory handle, the server connection ID, and a pointer to the directory path.

*driveNum* passes the drive number.

*newDirHandle* passes a pointer to the structure allocated for the new directory handle.

*effectiveRightsMask* passes a pointer to the space allocated for the trustee's effective rights to the directory connected via the newDirHandle parameter. See "Trustee Rights and Inherited Rights Mask" in Appendix A.

### Output

*newDirHandle* receives the new directory handle.

*effectiveRightsMask* receives the client's effective rights to the directory connected via the newDirHandle parameter. See "Trustee Rights and Inherited Rights Mask" in Appendix A.

### Return Values

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

0x98    Volume Does Not Exist  
0x9C    Invalid Path

See Appendix B for a complete listing of possible NetWare errors.

### Description

The directory handles allocated by this call are automatically deallocated when either the client logs out (or is somehow terminated), a call is made to NWDeallocateDirHandle or the dirHandle is overwritten by another allocation. The client should keep track of allocated directory handles and the directories they represent.

The driveNum parameter should be a unique number between 1-32. This number is required by the server when allocating dirHandles. The number could be used by the client in a table to represent a particular dirHandle, although the actual dirHandle must always be used when sending requests to the server.

The effectiveRightsMask receives the client's effective rights to the directory as determined by ANDing the directory's Inherited Rights Mask and the client's trustee rights in that directory.

## Notes

If a volume is specified in the path name, the volume must be represented by the volume name followed by a colon. You should pass a different drive number for each dirHandle allocated to avoid overwriting an existing dirHandle.

## See Also

NWAllocPermanentDirHandle  
NWDeallocateDirHandle  
NWParseFullPath

## NWDeallocateDirHandle

This function deallocates an allocated directory handle.

## Synopsis

```
#include "nwapi.h"

int                ccode;
uint16            serverConnID;
NWDirHandle_ts    dirHandle;

ccode=NWDeallocateDirHandle( serverConnID, dirHandle );
```

## Input

*serverConnID* passes the file server connection ID.

*dirHandle* passes the directory handle to be deallocated.

## Output

None.

## Return Values

0	Successful.
-1	Unsuccessful.

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

## Notes

This API does not delete the directory handle. The API breaks the connection so that the directory handle does not point anywhere.

When a client terminates, logs out or somehow loses its connection, all directory handles for that workstation are deleted. An End-Of-Job also deallocates temporary directory handles.

## See Also

NWAllocPermanentDirHandle  
NWAllocTemporaryDirHandle  
NWGetDirPath  
NWParseFullPath

## NWGetDirPath

This function returns the path name of the directory to which the given directory handle is associated.

Synopsis

```
#include "nwapi.h"

int          ccode;
uint16      serverConnID;
NWDirHandle_ts  dirHandle;
char        dirPath[NWMAX_DIR_PATH_LENGTH];

ccode=NWGetDirPath( serverConnID, dirHandle, dirPath );
```

Input

*serverConnID* passes the file server connection ID.

*dirHandle* passes the directory handle for the directory whose path is to be reported.

*dirPath* passes a pointer to the space allocated for the directory path name associated with the directory handle (above).

Output

*dirPath* receives the directory path name associated with the directory handle (above).

Return Values

- 0        Successful.
- 1      Unsuccessful.    One of the following error codes is placed in NWErrno:
  - 0x9C    Invalid Path
  - 0x9B    Bad Directory Handle
  - 0xFF    Path Not Locatable
  - 0xFB    Invalid Parameters

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

Description

This function returns the full path to the directory specified by the given directory handle.

See Also

- NWAllocTemporaryDirHandle
- NWParseFullPath

NWParseFullPath

This function parses a directory path string.

Synopsis

```
#include "nwapi.h"

int          ccode;
char         path[NWMAX_DIR_PATH_LENGTH];
char         server[NWMAX_SERVER_NAME_LENGTH];
uint16      serverConnID;
char         volName[NWMAX_VOLUME_NAME_LENGTH];
char         directories[NWMAX_DIR_PATH_LENGTH];
```

```
ccode=NWParseFullPath( path, server, &serverConnID, volName,
directories );
```

## Input

*path* passes a pointer to the string containing the path to be parsed.

*server* passes a pointer to a string allocated for the server name.

*serverConnID* passes a pointer to the structure containing the serverConnID.

*volName* passes a pointer to the string allocated for the name of the volume.

*directories* passes a pointer to the string allocated for the directory names.

## Output

*server* receives the server name.

*serverConnID* receives the file server connection ID.

*volName* receives the name of the volume.

*directories* receives the directory names.

## Return Values

0	Successful.
-1	Unsuccessful. One of the following error codes is placed in NWErrno:
0x9C	Invalid Path
0x9B	Bad Directory Handle
0xFF	Path Not Locatable
0xFB	Invalid Parameters

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

## Description

The path parameter must contain a full path name.

If the path is on a local drive, an error is returned. If the path specifies a file server name and there are no connections to that file server, the NO\_CONNECTIONS error is returned.

## See Also

NWGetDirPath

## NWSetDirHandle

This function sets the current directory for the specified directory handle.

## Synopsis

```
#include "nwapi.h"
```

```
int                                ccode;
NWPath_t                          path;
```

```
NWDirHandle_ts      targetDirHandle;  
  
ccode=NWSetDirHandle( &path, targetDirHandle );
```

## Input

*path* passes a pointer to the NWPath\_t structure created for the directory handle, the server connection ID and a pointer to the directory path.

*targetDirHandle* passes the target directory handle that becomes the new directory handle for the specified directory.

## Output

None.

## Return Values

0	Successful.
-1	Unsuccessful.

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

## Description

This function assigns the target directory handle to a directory path defined by the combined source directory handle and the source directory path.

The dirHandle from the NWPath\_t structure is an index number from 1 to 255. The dirHandle points to a volume or a directory on the file server. A file server maintains a Directory Handle table for each logged in client. If this call fails, the targetDirHandle parameter remains unchanged.

This call can only change dirHandles among directories on the same file server. In cases where multiple file servers are being used, the dirHandle and targetDirHandle parameter must have the same server connection ID.

The pathName parameter from the NWPath\_t structure can identify a full or partial directory path. A full directory path defines a volume or a directory on a given file server in the format VOLUME:DIRECTORY/.../DIRECTORY. A partial directory path specifies at least a directory, and possibly one or more parent directories.

Applications frequently combine a directory handle and a directory path to specify a target directory. For example, if the specified directory handle points to SYS: and the specified directory path = PUBLIC/WORDP, then the specified directory is SYS:PUBLIC/WORDP.

When an application defines a target directory using only a directory handle, the application must pass a null string in the pathName parameter. When an application defines a directory using only a directory path, the application must set dirHandle to zero.

## Notes

Directory handles must be kept track of separately for each file server connection.

## See Also

NWParseFullPath  
NWGetDirPath