Appendix 5A
FileDir API

Contents

Device-Specific APIs

DEVDeregisterDevice 4
DEVFindDeviceHandle 5
DEVRegisterDeviceName 6

Directory-Specific APIs

DIRAllocDirHandle 7
DIRCloseSearch 9
DIRDelete 10
DIRDup 11
DIREnumerateDirs 12
DIRFreeDirHandle 13
DIRGetAttributes 14
DIRGetDirectory 15
DIRGetDirectorySpace 16
DIRGetEntryInfo 17
DIRGetVolumeID 18
DIRGetVolumeInfo 19
DIRGetVolumeName 21
DIRMakeDirectory 22
DIRNextSearch 23
DIROpenSearch 24
DIRRedoSearch 26
DIRRename 27
DIRSetAttributes 28
DIRSetDirectory 30
DIR32To8Bit 31
DIR8To32Bit 32

File-Specific APIs

FILEAbort 33
FILEBuildFIB 34
FILEClose 35
FILECommit 36
FILEDup 37
FILEFindFIB 38
FILEGetDateTime 39
FILEGetInfo 40
FILEGetSize 41

FILEOpenCreate 42
FILERead 44
FILERemoteCopy 45
FILESeek 46
FILESetDateTime 47
FILEWrite 48

Synchronization-Specific APIs

SYNCCloseSemaphore 49
SYNCExamineSemaphore 50
SYNCFileName 51
SYNCFileSet 52
SYNCLogicalRecord 53
SYNCLogicalRecordSet 54
SYNCOpenSemaphore 55
SYNCPhysRecord 58
SYNCPhysRecordSet 60
SYNCSignalSemaphore 61
SYNCWaitOnSemaphore 62

DEVDeregisterDevice

Description

Deregisters a device from the parsing exclusion list.

```
UINT32
DEVDeregisterDevice (
    UINT32    pgID,
    UINT32    processID,
    UINT32    devHandle)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

devHandle Alias device handle.

Output

Returns result value.

Return values

None.

DEVFindDeviceHandle

Description Finds a device handle by matching device name

Syntax

```
UINT32
DEVFindDeviceHandle (
    UINT32     pgID,
    UINT32     processID,
    UINT8  *deviceName,
    UINT32     *foundDevHandle)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

deviceName Address of device name (<260 bytes).

Output

foundDevHandle Addresss to store alias device handle, zero if not found.

Return values None.

DEVRegisterDeviceName

Description

Registers a device name for exclusion from parsing.

Syntax

```
UINT32
DEVRegisterDeviceName
    ModHdlP         modHandle,
    UINT32          pgID,
    UINT32          processID,
    CONN_HANDLE     connHandle,
    UINT32          deviceType,
    UINT8           *deviceName,
    UINT32          *devHandle)
```

Input

modHandle Module handle of the calling NLM.

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

connHandle Connection handle or 0xFFFFFFFF if none.

deviceType Flag denoting device type (see Appendix 5B 'DEVICE_TYPE').

deviceName Address of device name (<260 bytes).

Output

devHandle Address to return alias device handle.

Return values

None.

DIRAllocDirHandle

Description

Allocates a directory handle to specified path.

Syntax

```
UINT32
DIRAllocDirHandle (
    ModHdlP   modHandle,
    UINT32    pgID,
    UINT32    processID,
    UINT8     typeFlag,
    UINT8     dirTag,
    UINT32    srcDirHandle,
    UINT8     nameSpace,
    UINT8     *path,
    UINT32    *newDirHandle)
```

Input

modHandle Module handle of the calling NLM.

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

typeFlag Bits set as follows:
0x01 lives beyond application, otherwise terminates with application
0x02 rooted at path mapping, otherwise rooted at volume.

dirTag Name to be associated with new directory handle.

srcDirHandle Directory handle that the path is relative to, or NULL if path fully specified.

nameSpace Name space type (see Appendix 5B 'NAME_SPACE'...).

path Address of a buffer holding the path relative to the supplied directory handle.

newDirHandle Address of a buffer to store the new directory handle.

Output

Returns result value.

Return values

SUCCESS_CODE	
INVALID_DIR_HANDLE	*srcDirHandle* is invalid
INVALID_PARAMETER	*typeFlag* is invalid or *processID* not specified when the typeFlag bit 0 is clear
OUT_OF_CLIENT_MEMORY	Not enough memory to satisfy request.

See NCPERROR.H for list of NCP codes

DIRCloseSearch

Description

Closes specified search context.

Syntax

```
UINT32
DIRCloseSearch (
    UINT32    pgID,
    UINT32    processID,
    UINT32    sibHandle)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

sibHandle Search info block handle obtained from **DIROpenSearch**.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_SEARCH_HANDLE if sib handle is invalid

See NCPERROR.H for list of NCP codes.

DIRDelete

Description

Deletes specified directory entry.

Syntax

```
UINT32
DIRDelete (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     nameSpace,
    UINT8     *path,
    UINT32    attributes)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Directory handle that the path is relative to, or NULL if path fully specified.

nameSpace Name space type (see Appendix 5B 'NAME_SPACE'...).

path Address of a buffer holding the path and filename.

attributes Attributes to search by.

Output

Returns result value.

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

DIRDup

Description

Returns a duplicate of the specified directory handle.

Syntax

```
UINT32
DIRDup (
    ModHdlP   modHandle,
    UINT32    srcPgID,
    UINT32    srcProcessID,
    UINT32    srcDirHandle,
    UINT32    newPgID,
    UINT32    newProcessID,
    UINT32    *newDirHandle)
```

Input

modHandle	Module handle of the calling NLM.
srcPgID	Process group ID of the owning process .
srcProcessID	Process ID of the calling process.
srcDirHandle	Directory handle to duplicate.
newPgID	Process group ID of the new process group.
newProcessID	Process ID of the calling process.
newDirHandle	Address of a buffer to store the new directory handle.

Output

Returns result value.

Return values

SUCCESS_CODE	
OUT_OF_CLIENT_MEMORY	Not enough memory to satisfy request.

See NCPERROR.H for list of NCP codes.

DIREnumerateDirs

Description

Returns directory mapping information.

Syntax

```
UINT32
DIREnumerateDirs (
    UINT32          pgID,
    UINT32          processID,
    CONN_HANDLE     connHandle,
    UINT32          *searchIndex,
    DirMapInfo      *dmap)
```

Input

pgID Process group ID to match, or 0xFFFFFFFF for any.

processID Process ID of the calling process, or 0xFFFFFFFF for any.

connHandle Connection handle to match, or 0xFFFFFFFF for any.

searchIndex Address of buffer to hold search index; this must be 0xFFFFFFFF to start and will be modified for subsequent calls to enumerate through all directories.

dmap Address of buffer to hold dir map information in the following format:

```
UINT32      DirHandle32;
UINT32      VolumeID;
CONN_HANDLE ConnHandle;
UINT32      AliasHandle;
ModHdlP     ModHandle;
UINT32      PG_ID;
UINT32      ProcessID;
UINT32      FakeRootDepth;
UINT32      CDDepth;
UINT16      Flags;  (DIB_FLAG_...)
UINT8       DirHandle8;  (0 if none)
UINT8       DirHandleName;
UINT8       NameSpace;
```

Output

Returns result value.

Return values

SUCCESS_CODE
NO_MORE_ENTRIES if search has been exhausted

DIRFreeDirHandle

Description

Deallocates the specified directory handle.

Syntax

```
UINT32
DIRFreeDirHandle (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Directory handle that the path is relative to, or NULL if path fully specified.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_DIR_HANDLE if dirHandle is invalid.

See NCPERROR.H for list of NCP codes.

DIRGetAccessRights

Description

Gets access rights for the specified directory.

Syntax

```
UINT32
DIRGetAccessRights
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     nameSpace,
    UINT8     *path,
    UINT32    *accessRights)
```

Input

pgID Process Group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Alias directory handle, zero if path is fully specified.

nameSpace Name space type. See Appendix 5B under "NAME_SPACE".

path Address of input path.

Output

accessRights Address to return access rights. (See Appendix 5B, "RIGHTS_".)

Return values

SUCCESS_CODE

Remarks

See Appendix 3C for list of NCP codes.

DIRGetAttributes

Description

Gets entry's attributes.

Syntax

```
UINT32
DIRGetAttributes (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     nameSpace,
    UINT8     *path,
    UINT32    *attributes)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Directory handle that the path is relative to, or NULL if path fully specified.

nameSpace Name space type (see Appendix 5B 'NAME_SPACE'...).

path Address of a buffer holding the path and filename.

attributes Address of a buffer to store the file entry's attributes.

Output

attributes File entry's attributes.

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

DIRGetDirectory

Description

Returns UNC path for a directory handle and relative path.

Syntax

```
UINT32
DIRGetDirectory (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     nameSpace,
    UINT8     *applyPath,
    UINT8     *qualifiedPath)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Directory handle that the path is relative to, or NULL if path fully specified.

nameSpace Name space type (see Appendix 5B 'NAME_SPACE'...).

applyPath Address of a buffer to apply to the path obtained from the *dirHandle*.

qualifiedPath Address of a buffer to store the resultant path merge of the dirHandle path and passed applyPath where the root path is null-terminated followed by the relative path null-terminated. If there is no relative path, then the root path is double-null-terminated.

Output

qualifiedPath Buffer is filled.

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

DIRGetDirectorySpace

Description

Gets directory space information.

Syntax

```
UINT32
DIRGetDirectorySpace (
    UINT32      pgID,
    UINT32      processID,
    UINT32      dirHandle,
    DiskSpace *diskSpace)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
dirHandle	Directory handle.
diskSpace	Address of buffer to store disk space information.

Output

totalSpace	Filled out accordingly (this is part of the diskSpace structure).
freeSpace	Filled out accordingly (this is part of the diskSpace structure).

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

DIRGetEntryInfo

Description

Gets an entry's information

Syntax

```
UINT32
DIRGetEntryInfo
    UINT32      pgID,
    UINT32      processID,
    UINT32      dirHandle,
    UINT8  nameSpace,
    UINT8  *path,
    DirEntryInfo     *dEntry)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
dirHandle	Alias directory handle, zero if path is fully specified.
nameSpace	Name space type (See Appendix B, 'NAME_SPACE')
path	Address of input path.

Output

dEntry	Address to return information (see DirEntryInfo in Appendix 5B).

Return values

None.

DIRGetMapInfo

Description

Returns directory mapping information.

Syntax

```
UINT32
DIRGetMapInfo
    UINT32      pgID,
    UINT32      processID,
    UINT32      dirHandle,
    DirMapInfo  *dmap)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
dirHandle	Alias directory handle.

Output

dmap	Address to return directory mapping information (see 'DirMapInfo' in CLIENT32.H for format).

Return values

SUCCESS_CODE
INVALID_DIR_HANDLE

DIRGetVolumeID

Description

Gets volume ID.

Syntax

```
UINT32
DIRGetVolumeID (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     *path,
    UINT32    *volumeID)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
dirHandle	Directory handle to which the path is relative, or NULL if path is fully specified.
path	Address of a buffer holding the path relative to the supplied directory handle.
volumeID	Buffer to store volume ID.

Output

Returns result value.

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

DIRGetVolumeInfo

Description

Gets volume information.

Syntax

```
UINT32
DIRGetVolumeInfo (
    UINT32    pgID,
    UINT32    processID,
    UINT8     *serverVolume,
    UINT32    *flags,
    UINT32    *maxFileName,
    UINT32    *maxPath,
    UINT8     *fileSystemName)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
serverVolume	Null-terminated address of "\\SERVER\VOLUME".
flags	Address of buffer to receive "FS..." flags (see Appendix 5B for definitions).
maxFileName	Address of buffer to receive maximum file-name length supported by this volume.
maxPath	Address of buffer to receive maximum path-name length supported by this volume.
fileSystemName	Address of buffer to receive the name of the file system.

Output

flags	Filled out accordingly.
maxFileName	Filled out accordingly.
maxPath	Filled out accordingly.
fileSystemName	Filled out accordingly.

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

DIRGetVolumeName

Description

Gets volume name.

Syntax

```
UINT32
DIRGetVolumeName (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     *volumeName)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Alias directory handle.

volumeName Buffer to store NULL-terminated volume name (up to 17 bytes including NULL).

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_DIR_HANDLE Invalid *dirHandle* specified.

See NCPERROR.H for list of NCP codes.

DIRMakeDirectory

Description

Creates the specified directory.

Syntax

```
UINT32
DIRMakeDirectory (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     nameSpace,
    UINT8     *path)
```

Input

pgID Process group ID of the calling process group

processID Process ID of the calling process.

dirHandle Directory handle that the path is relative to, or NULL if path fully specified.

nameSpace Name space type (see Appendix 5B 'NAME_SPACE'...).

path Holds address of a buffer holding the path and directory name.

Output

Returns result value.

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

DIRNextSearch

Description

Returns directory entry information.

Syntax

```
UINT32
DIRNextSearch (
    UINT32          pgID,
    UINT32          processID,
    UINT32          sibHandle,
    DirEntryInfo *dEntry)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

sibHandle Search information block handle obtained from **DIROpenSearch**.

dEntry Buffer to store directory entry information in the following format:

```
UINT16   Reserved;
UINT8    Name[14];
UINT16   Attributes;
UINT32   SizeLo;
UINT32   SizeHi;
UINT32   CreatorsID;
UINT32   ModifiersID;
UINT32   ArchiversID ;
UINT16   CreationDate;
UINT16   CreationTime;
UINT16   AccessDate;
UINT16   UpdateDate;
UINT16   UpdateTime;
UINT8    LongName[MAX_PATH_LENGTH];
```

Output

dEntry Directory information filled in.

Return values

SUCCESS_CODE
INVALID_SEARCH_HANDLE Sib handle is invalid.

See NCPERROR.H for list of NCP codes.

DIROpenSearch

.

Description

Returns search context for specified filespec.

Syntax

```
UINT32
DIROpenSearch (
    ModHdlP modHandle,
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     nameSpace,
    UINT8     *path,
    UINT32    attributes,
    UINT32    showDotsFlag,
    UINT32    *sibHandle)
```

Input

modHandle	Module handle of the calling NLM.
pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
dirHandle	Directory handle to which the path is relative, or NULL if path is fully specified.
nameSpace	Name space type .
path	Address of a buffer holding the path and filename.
attributes	Desired search attributes.
showDotsFlag	One to show dots ("." and ".." if search mask allows) or zero for no dots.
sibHandle	Address of a buffer to store the search info block handle.

Output	*dsc*	Filled out for next search.

Return values

SUCCESS_CODE	
OUT_OF_CLIENT_MEMORY	Not enough memory to satisfy request.

See NCPERROR.H for list of NCP codes.

DIRRedoSearch

Description

Reinitializes search context to new attributes.

Syntax

```
UINT32
DIRRedoSearch (
    UINT32    pgID,
    UINT32    processID,
    UINT32    sibHandle,
    UINT32    attributes)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
sibHandle	Search information block handle obtained from **DIROpenSearch**.
attributes	New desired search attributes.

Output

Returns result value.

Return values

SUCCESS_CODE	
INVALID_SEARCH_HANDLE	SIB handle is invalid.

See NCPERROR.H for list of NCP codes.

DIRRename

Description

Renames specified directory entry.

Syntax

```
UINT32
DIRRename (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     nameSpace,
    UINT8     *srcPath,
    UINT8     *destPath,
    UINT32    attributes)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Directory handle that the path is relative to, or NULL if path fully specified.

nameSpace Name space type (see Appendix 5B 'NAME_SPACE'...).

srcPath Address of a buffer holding the path and filename being renamed.

destPath Address of a buffer holding the path and filename of new name.

attributes Attributes to search by (file or directory).

Output

Returns result value.

Return values

SUCCESS_CODE
NOT_SAME_DEVICE Trying to copy across different servers.

See NCPERROR.H for list of NCP codes.

DIRSetAliasObjectID

Description Sets Alias Object ID for the dirHandle

Syntax

```
UINT32
DIRSetAliasObjectID
    UINT32 pgID,
    UINT32 processID,
    UINT32 dirHandle,
    UINT32 aliasObjectID)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
dirHandle	Alias directory handle.

Output

aliasObjectID	Alias object ID.

Return values

SUCCESS_CODE
INVALID_DIR_HANDLE

DIRSetAttributes

Description Sets entry's attributes.

Syntax

```
UINT32
DIRSetAttributes (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     nameSpace,
    UINT8     *path,
    UINT32    attributes,
    UINT32    applyMask)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Directory handle that the path is relative to, or NULL if path fully specified.

nameSpace Name space type (see Appendix 5B 'NAME_SPACE'...).

path Address of a buffer holding the path and filename.

attributes File entry's attributes to set (see ATTR_ in Appendix 5B).

applyMask Mask indicating which bits to modify.

Output Returns result value.

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

Remarks

The archive bit may be set regardless of access rights. Requests to set the directory or volume bit are ignored.

DIRSetDirectory

Description

Sets directory handle to new path.

Syntax

```
UINT32
DIRSetDirectory (
    UINT32    pgID,
    UINT32    processID,
    UINT32    dirHandle,
    UINT8     nameSpace,
    UINT8     *path)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Directory handle that the path is relative to.

nameSpace Name space type (see Appendix 5B 'NAME_SPACE'...).

path Address of a buffer holding the path.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_DIR_HANDLE

See NCPERROR.H for list of NCP codes.

DIR32To8Bit

Description

Converts a 32-bit directory handle to 8-bit.

Syntax

```
UINT32
DIR32To8Bit (
    UINT32      pgID,
    UINT32      processID,
    UINT32      dirHandle,
    CONN_HANDLE *connHandle,
    UINT8       *aliasHandle8)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
dirHandle	32-bit directory handle.
connHandle	Buffer to store the connection handle.
aliasHandle8	Address of buffer to store 8-bit directory handle.

Output

aliasHandle8	Filled out accordingly.

Return values

SUCCESS_CODE	
INVALID_DIR_HANDLE	*dirHandle* is invalid
OUT_OF_CLIENT_MEMORY	Insufficient memory to satisfy request

DIR8To32Bit

Description

Converts an 8-bit directory handle to 32-bit.

Syntax

```
UINT32
DIR8To32Bit (
    UINT32          pgID,
    UINT32          processID,
    CONN_HANDLE connHandle,
    UINT8           aliasHandle8,
    UINT32          *dirHandle)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
connHandle	Connection handle to match in a search.
aliasHandle8	8-bit directory handle.
dirHandle	Address of buffer to store 32-bit directory handle.

Output

dirHandle	Filled out accordingly.

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

FILEAbort

Description Cleans up file entry without accessing network.

Syntax

```
UINT32
FILEAbort (
    UINT32    pgID,
    UINT32    processID,
    UINT32    fibHandle)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
fibHandle	Alias file handle.

Output Returns result value.

Return values None.

FILEBuildFIB

Description

Returns FIB handle for specified file description.

Syntax

```
UINT32
FILEBuildFIB (
    ModHdlP         modHandle,
    UINT32              pgID,
    UINT32              processID,
    CONN_HANDLE     connHandle,
    BuildFIB            *bfib,
    UINT32              *fibHandle)
```

Input

modHandle Module handle of the calling NLM.

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

connHandle Connection handle.

bfib File description structure as follows:

```
                              Must fill
UINT8    FileHandle[6];       yes
UINT16   Reserved;            0
UINT8    NameZ[14];           yes
UINT8    AccessRights;        use NW_FACC
UINT8    Reserved2;           0
UINT32   Size;                actual or 0
UINT16   CreationDate;        actual or 0
UINT16   LastAccessDate;      actual or 0
UINT16   LastUpdateDate;      actual or 0
UINT16   LastUpdateTime;      actual or 0
UINT16   CreationTime;        actual or 0
```

fibHandle Address to store the alias file handle.

Output

nwHandle NetWare file handle.

Return values

SUCCESS_CODE	
INVALID_FILE_HANDLE	
OUT_OF_CLIENT_MEMORY	Not enough memory to satisfy request.

FILEClose

Description

Closes specified file.

Syntax

```
UINT32
FILEClose (
    UINT32    pgID,
    UINT32    processID,
    UINT32    fibHandle)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

fibHandle Alias file handle.

Output

Returns result value.

Return values

SUCCESS_CODE or 1 for not the last open
INVALID_FILE_HANDLE

See NCPERROR.H for list of NCP codes.

FILECommit

Description

Commits specified file's dirty write buffers

Syntax

```
UINT32
FILECommit (
    UINT32    pgID,
    UINT32    processID,
    UINT32    fibHandle)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

fibHandle Alias file handle.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_FILE_HANDLE

See NCPERROR.H for list of NCP codes.

FILEConnCheck

Description Determines if any files are open on a connection.

Syntax

```
BOOL
FILEConnCheck (
    CONN_HANDLE   connHandle)
```

Input

connHandle Connection handle to check for.

Return values TRUE or FALSE

FILEDup

Description

Increments the count of times opened.

Syntax

```
UINT32
FILEDup (
    UINT32    pgID,
    UINT32    parentPID,
    UINT32    childPID,
    UINT32    fibHandle)
```

Input

pgID	Process group ID of the calling process group.
parentPID	Process ID of the parent process.
childPID	Process ID of the child process, or is the same as the parent process.
fibHandle	Alias file handle.

Output

Returns result value.

Return values

SUCCESS_CODE	
INVALID_FILE_HANDLE	
NO_MORE_ENTRIES	Number of supported children is exhausted.

See NCPERROR.H for list of NCP codes.

FILEFindFIB

Description

Finds FIB handle by connHandle and 6-byte file handle.

Syntax

```
UINT32
FILEFindFIB (
    UINT32          pgID,
    UINT32          processID,
    CONN_HANDLE     connHandle,
    UINT8           *fileHandle,
    UINT32          *fibHandle)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
connHandle	Connection handle associated with file handle.
fileHandle	Address of 6 byte file handle.
fibHandle	Address of buffer to hold returned alias file handle.

Output

fibHandle	Filled accordingly.

Return values

SUCCESS_CODE	
INVALID_FILE_HANDLE	Handle not found

FILEGetDateTime

Description

Returns file's date and time.

Syntax

```
UINT32
FILEGetDateTime (
    UINT32      pgID,
    UINT32      processID,
    UINT32      fibHandle,
    NDateTime   *dateTime)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
fibHandle	Alias file handle.
dateTime	Pointer to store the dateTime structure (see NIOS.H for NDateTime).

Output

**dateTime*	Filled accordingly.

Return values

SUCCESS_CODE
INVALID_FILE_HANDLE

FILEGetInfo

Description

Returns information about a file handle.

Syntax

```
UINT32
    UINT32      pgID,
    UINT32      processID,
    UINT32      fibHandle,
    FileInfo    *fileInfo)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
fibHandle	Alias file handle.
fileInfo	Buffer to store the FileInfo structure.

Output

fileInfo	Filled accordingly

Return values

SUCCESS_CODE
INVALID_FILE_HANDLE

FILEGetSize

Description

Returns the current file size.

Syntax

```
UINT32
FILEGetSize (
    UINT32    pgID,
    UINT32    processID,
    UINT32    fibHandle,
    UINT64    *fileSize)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

fibHandle Alias file handle.

fileSize Address to store file size.

Output

fileSize File size.

Return values

SUCCESS_CODE
INVALID_FILE_HANDLE

See NCPERROR.H for list of NCP codes.

FILEOpenCreate

Description

Opens or creates the specified file.

Syntax

```
UINT32
FILEOpenCreate (
        ModHdlP modHandle,
        UINT32    pgID,
        UINT32    processID,
        UINT32    dirHandle,
        UINT8     nameSpace,
        UINT8     *path,
        UINT32    openMode,
        UINT32    actionBits,
        UINT32    attributes,
        UINT32    *fibHandle)
```

Input

modHandle Module handle of the calling NLM.

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

dirHandle Directory handle that the path is relative to, or NULL if path fully specified.

nameSpace Name space type (see Appendix 5B 'NAME_SPACE'...).

path Address of a buffer holding the path and filename.

openMode Bits that match DOS open mode access bits. See "ACCESS_" in Appendix 5B for equates.

actionBits Bits that match DOS open/create action bits.
0 = Fail, 1 = Create,
bits 1,2 indicate action if exists,
00 = Fail, 01 = Open, 10 = Create, 11 = Invalid.
See "ACTION_" in Appendix 5B for equates.

attributes Low byte used to assign attributes when creating the file; otherwise, should be zero.

	fibHandle	Address to store the alias file handle.
Output	*fibHandle*	Stored alias file handle, or zero if failed.

Return values

NCP_NO_MORE_FILE_HANDLES	No more memory to allocate FIBs
OUT_OF_CLIENT_MEMORY	Not enough memory to satisfy request

See NCPERROR.H for list of NCP codes.

FILERead

Description

Reads specified file from cache or network.

Syntax

```
UINT32
FILERead (
    UINT32    pgID,
    UINT32    processID,
    UINT32    fibHandle,
    UINT32    ioLength,
    UINT8     *ioData,
    UINT32    *ioCompleted)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
fibHandle	Alias file handle.
ioLength	Amount to read.
ioData	Pointer to linear memory of data to read.
ioCompleted	Pointer to store number of bytes read.

Output

**ioCompleted*	Number of bytes read.

Return values

SUCCESS_CODE data was read
INVALID_FILE_HANDLE

See NCPERROR.H for list of NCP codes.

FILERemoteCopy

Description

Copies data from one file to another.

Syntax

```
UINT32
FILERemoteCopy (
    UINT32    pgID,
    UINT32    processID,
    UINT32    fibHandleSrc,
    UINT32    fibHandleDest,
    UINT32    bytesToCopy,
    UINT32    *bytesCopied)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
fibHandleSrc	Source alias file handle.
fibHandleDest	Destination alias file handle.
bytesToCopy	Number of bytes to copy from/to current seek positions.

Output

Both *fib->SeekPositionLo*'s have updated positions

Return values

SUCCESS_CODE
INVALID_FILE_HANDLE

See NCPERROR.H for list of NCP codes.

FILESeek

Description

Sets current file position.

Syntax

```
UINT32
FILESeek (

    UINT32      pgID,
    UINT32      processID,
    UINT32      fibHandle,
    SINT64      seekOffset,
    UINT32      flagOrigin,
    UINT64      *newPosition)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
fibHandle	Alias file handle.
seekOffset	Signed long integer offset from origin.
flagOrigin	Origin of move: 0 - SEEK_FROM_START 1 - SEEK_FROM_CURRENT 2 - SEEK_FROM_END.
newPosition	Buffer to store updated position.

Output

newPosition	Updated position.

Return values

SUCCESS_CODE
INVALID_FILE_HANDLE
INVALID_PARAMETER

See NCPERROR.H for list of NCP codes.

FILESetDateTime

Description Sets file's date and time.

Syntax

```
UINT32
FILESetDateTime (
    UINT32      pgID,
    UINT32      processID,
    UINT32      fibHandle,
    NDateTime   *dateTime)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
fibHandle	Alias file handle.
dateTime	Pointer to dateTime structure (see NIOS.H).

Output

dateTime	Filled accordingly.

Return values

SUCCESS_CODE
INVALID_FILE_HANDLE

FILEWrite

Description

Writes specified file to cache or network.

Syntax

```
UINT32
FILEWrite (
    UINT32    pgID,
    UINT32    processID,
    UINT32    fibHandle,
    UINT32    ioLength,
    UINT8     *ioData,
    UINT32    *ioCompleted)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
fibHandle	Alias file handle.
ioLength	Amount to write.
ioData	Pointer to linear memory of data to write.
ioCompleted	Pointer to store number of bytes written.

Output

**ioCompleted*	Number of bytes written

Return values

SUCCESS_CODE
INVALID_FILE_HANDLE

See NCPERROR.H for list of NCP codes.

SYNCCloseSemaphore

Description Closes the specified semaphore.

Syntax

```
UINT32
SYNCCloseSemaphore (
    UINT32    pgID,
    UINT32    processID,
    UINT32    semHandle)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

semHandle Alias semaphore handle returned from the open.

Output Returns result value.

Return values SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

SYNCExamineSemaphore

Description

Examines the current count of a semaphore.

Syntax

```
UINT32
SYNCExamineSemaphore (
    UINT32    pgID,
    UINT32    processID,
    UINT32    semHandle,
    UINT32    *curOpenCount,
    UINT32    *value)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

semHandle Alias semaphore handle returned from the open.

curOpenCount Number of clients that currently have the semaphore open.

value Address to store the current semaphore value.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_SEM_HANDLE *semHandle* is invalid.

See NCPERROR.H for list of NCP codes.

Remarks

A positive value indicates that the caller can access theresource. A negative value indicates the caller must wait for the semaphore by calling **SYNCWaitOnSemaphore,** or must temporarily abandon the operation. **SYNCSignalSemaphore** increments count. **SYNCWaitOnSemaphore** decrements count.

SYNCFileName

Description

Provides services for file-based semaphores.

Syntax

```
UINT32
SYNCFileName (
    ModHdlP modHandle,
    UINT32  pgID,
    UINT32  processID,
    UINT32  dirHandle,
    UINT8   *path,
    UINT32  syncType,
    UINT32  tickTimeout)
```

Input

modHandle	Module handle of the calling NLM.
pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
dirHandle	Directory handle to which the path is relative, or NULL if path fully specified.
path	Address of a buffer holding the path and filename.
syncType	Type of synchronization to perform (see Appendix 5B for "SYNC_TYPE_..." equates).
tickTimeout	Number of 1/18 second ticks before timing out; zero means use default.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_PARAMETER *SyncType* is invalid.

See NCPERROR.H for list of NCP codes.

SYNCFileSet

Description

Sets all logged file-based semaphores active.

Syntax

```
UINT32
SYNCFileSet (
    UINT32    pgID,
    UINT32    processID,
    UINT32    syncType,
    UINT32    tickTimeout)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
syncType	Type of syncronization to perform (see Appendix 5B for "SYNC_TYPE_..." equates).
tickTimeout	Number of 1/18 second ticks before timing out; 0 means use default.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_PARAMETER *syncType* is invalid.

See NCPERROR.H for list of NCP codes.

SYNCLogicalRecord

Description

Provides services for string-based semaphores.

Syntax

```
UINT32
SYNCLogicalRecord (
    ModHdlP         modHandle,
    UINT32          pgID,
    UINT32          processID,
    CONN_HANDLE connHandle,
    UINT8           *syncName,
    UINT32          syncType,
    UINT32          tickTimeout)
```

Input

modHandle Module handle of the calling NLM.

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

connHandle Connection handle.

syncName Address of synchronization string (up to 128 bytes).

syncType Type of synchronization to perform (see Appendix 5B for "SYNC_TYPE_..." equates).

tickTimeout Number of 1/18 second ticks before timing out; 0 means use default.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_PARAMETER *SyncType* is invalid.

See NCPERROR.H for list of NCP codes.

SYNCLogicalRecordSet

Description

Sets active all logged string-based semaphores.

Syntax

```
UINT32
SYNCLogicalRecordSet (
    UINT32    pgID,
    UINT32    processID,
    UINT32    syncType,
    UINT32    tickTimeout)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

syncType Type of synchronization to perform (see Appendix 5B for "SYNC_TYPE_..." equates).

tickTimeout Number of 1/18 second ticks before timing out; zero means use default.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_PARAMETER *SyncType* is invalid.

See NCPERROR.H for list of NCP codes.

SYNCOpenSemaphore

Description

Opens or creates a named semaphore.

Syntax

```
UINT32
SYNCOpenSemaphore (
    ModHdlP modHandle,
    UINT32     pgID,
    UINT32     processID,
    CONN_HANDLE   connHandle,
    SINT32     initialValue,
    UINT32     nameLength,
    UINT8      *semaphoreName,
    UINT32     *semHandle,
    UINT32     *curOpenCount)
```

Input

modHandle	Module handle of the calling NLM.
pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
connHandle	Connection handle.
initialValue	Zero-based number of applications that can access the semaphore simultaneously (used only if the semaphore is being created).
nameLength	Length of the semaphore name.
semaphoreName	Address to retrieve the semaphore name.
semHandle	Address to store the semaphore handle.
curOpenCount	Address to store the current open count.

Output

semHandle	Filled out accordingly.
curOpenCount	Filled out accordingly.

Return values

SUCCESS_CODE
OUT_OF_CLIENT_MEMORY

See NCPERROR.H for list of NCP codes.

Remarks

Like a logical record lock, a semaphore is a name associated with network resources such as files, records, or structures. Both logical record locks and semaphores limit the number of applications that can access network resource(s) at one time. Logical record locks limit to one the number of applications that can access the resource. Semaphores, on the other hand, allow a configurable number of applications to access a network resource at one time.

When an application creates a semaphore, the application assigns a value to the semaphore (for example, 4). The value indicates how many applications can access the resource associated with the semaphore at one time. In the example, five applications can access the resource at one time (0 to 4).

After opening an existing semaphore, an application first checks the value using **SYNCExamineSemaphore**. If the value is greater than or equal to zero, the application can access the associated network resource. The application decrements the value by calling **SYNCWaitOnSemaphore** and then accesses the resource. When the application finishes accessing the resource, the application increments the semaphore value by calling **SYNCSignalSemaphore**, and then **SYNCExitSemaphore.**

If, after opening a semaphore, an application discovers that the value is negative, the application cannot access the resource immediately. The application can either wait a specified timeout interval until the resource becomes accessible, or the application can retry later.

The *currentOpenCount* indicates the number of processes using the semaphore. **SYNCOpenSemaphore** increments the count. **SYNCCloseSemaphore** decrements the count.

The following algorithm illustrates semaphore usage:

```
SYNCOpenSemaphore ()
SYNCExamineSemaphore ()
    If (semaphoreValue >= 0) {
        If ((SYNCWaitOnSemaphore ()) == 0) {
            Access the associated resource
            SYNCSignalSemaphore ()
        }
    }
SYNCCloseSemaphore
```

SYNCPhysRecord

Description

Provides services for file-region-based semaphores.

Syntax

```
UINT32
SYNCPhysRecord (
    UINT32    pgID,
    UINT32    processID,
    UINT32    fibHandle,
    UINT32    lockOffset,
    UINT32    lockLength,
    UINT32    syncType,
    UINT32    tickTimeout)
```

Input

pgID	Process group ID of the calling process group
processID	Process ID of the calling process.
fibHandle	Alias file handle.
lockOffset	Offset of the file to begin lock region.
lockLength	Length of lock region.
syncType	Type of synchronization to perform (see Appendix 5B for "SYNC_TYPE_..." equates).
tickTimeout	Number of 1/18 second ticks before timing out; 0 means use default.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_PARAMETER *syncType* is invalid

See NCPERROR.H for list of NCP codes.

Remarks

SYNC_TYPE_LOG	0x00	Logs only (lock can be set later)SYNC_TYPE_LOCK_EXCLUSIVE0x01 Logs and locks simultaneously
SYNC_TYPE_LOCK_SHARE	0x02	Logs and locks simultaneously
SYNC_TYPE_RELEASE	0x03	Releases lock
SYNC_TYPE_CLEAR	0x04	Releases lock and clears log

To obtain a fibHandle from a six-byte file handle, use **FILEFindFIB**. If none is found, use **FILEBuildFIB** to build one.

SYNCPhysRecordSet

Description

Sets active all logged file-region-based semaphores.

Syntax

```
UINT32
SYNCPhysRecordSet (
    UINT32    pgID,
    UINT32    processID,
    UINT32    syncType,
    UINT32    tickTimeout)
```

Input

pgID	Process group ID of the calling process group.
processID	Process ID of the calling process.
syncType	Type of synchronization to perform (see Appendix 5B for "SYNC_TYPE_..." equates).
tickTimeout	Number of 1/18 second ticks before timing out; zero means use default.

Output

Returns result value.

Return values

SUCCESS_CODE
INVALID_PARAMETER *SyncType* is invalid.

See NCPERROR.H for list of NCP codes.

SYNCSignalSemaphore

Description Signals the specified semaphore.

Syntax

```
UINT32
SYNCSignalSemaphore (
    UINT32    pgID,
    UINT32    processID,
    UINT32    semHandle)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

semHandle Alias semaphore handle returned from the open.

Output Returns result value.

Return values SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

SYNCWaitOnSemaphore

Description

Waits on the semaphore for the specified timeout.

Syntax

```
UINT32
SYNCWaitOnSemaphore (
    UINT32    pgID,
    UINT32    processID,
    UINT32    semHandle,
    UINT32    timeout)
```

Input

pgID Process group ID of the calling process group.

processID Process ID of the calling process.

semHandle Semaphore handle returned from the open.

timeout Time to retry in milliseconds (0 means no wait).

Output

Returns result value.

Return values

SUCCESS_CODE

See NCPERROR.H for list of NCP codes.

Remarks

The *timeout* parameter determines how long **WaitOnSemaphore** will wait before reincrementing the semaphore value and removing the requesting application from the queue.