LANtasticÒ Programmer's Reference

This document is a reference to the programmatic interface for Artisoft's LANtastic Network Operating System (NOS) Version 4.00. Some knowledge of either C or assembler, and the DOS conventions for system call interrrupts, is necessary for you to fully understand the information in this document.

Table Of Contents

µTable Of Contents 1
Calling Conventions 3
Assembly Language 3
Assembler Interface 3
C Language 3
C Language Interface 4
The LANtastic Interface 5
Network Pathnames 5
Error Codes And Error Handling 5
Configuration Testing 9
Testing For The Existence Of The NetBIOS 9
Testing For The Existence Of The Redirector, Server And LANPUP 9
Testing For The Existence Of SHARE 10
Determining The Network Software Version 10
Testing For The Alternate NetBIOS Interface - Interrupt 2AH 11
INT 2AH Function 0: INSTALLATION CHECK 11
INT 2AH Function 1: EXECUTE NetBIOS WITH NO ERROR RETRY 11
INT 2AH Function 5: GET RESOURCE INFORMATION 12
Controlling Redirected Printer Streams 13
Controlling Printers With Interrupt 2A 13
SET SPOOLED OUTPUT TO COMBINE MODE 13
SET SPOOLED OUTPUT IN SEPARATE MODE 13
FLUSH PRINTER OUTPUT 14
Controlling Printers With Interrupt 21 14
RETURN REDIRECTED PRINTER MODE 14
SET REDIRECTED PRINTER MODE 14
FLUSH PRINTER OUTPUT 15
DOS Compatible Network Functions 16
GET MACHINE NAME 16
SET PRINTER SETUP 17
GET PRINTER SETUP 18

GET REDIRECTED DEVICE ENTRY 18
REDIRECT DEVICE 19
CANCEL DEVICE REDIRECTION 20
NOS Extended Network Functions 21
GET LOGIN ENTRY 21
LOGIN TO A SERVER 22
LOGOUT OF A SERVER 22
GET USERNAME ENTRY 23
GET INACTIVE SERVER ENTRY 23
CHANGE PASSWORD 24
DISABLE ACCOUNT 24
GET ACCOUNT 25
LOGOUT FROM ALL SERVERS 26
COPY FILE 26
SEND UNSOLICITED MESSAGE 27
GET LAST RECEIVED UNSOLICITED MESSAGE 28
GET MESSAGE PROCESSING FLAG 28
SET MESSAGE PROCESSING FLAG 29
POP UP LAST RECEIVED MESSAGE 29
GET QUEUE ENTRY 30
SET QUEUE ENTRY 31
CONTROL QUEUE 31
GET PRINTER STATUS 32
GET STREAM INFO 33
SET STREAM INFO 34
CREATE USER AUDIT ENTRY 34
GET ACTIVE USER INFORMATION 35
GET SHARED DIRECTORY INFORMATION 36
GET USERNAME FROM ACCOUNT FILE 37
TRANSLATE PATH 37
CREATE INDIRECT FILE 38
GET INDIRECT FILE CONTENTS 38
GET SERVER'S TIME 39
SCHEDULE SERVER SHUTDOWN 39
CANCEL SERVER SHUTDOWN 40
STUFF SERVER KEYBOARD BUFFER 40
GET REDIRECTED PRINTER TIMEOUT 41
SET REDIRECTED PRINTER TIMEOUTS 41
GET DOS SERVICE VECTOR 42
SET DOS SERVICE VECTOR 42
GET MESSAGE SERVICE VECTOR 43
SET MESSAGE SERVICE VECTOR 43
Other System Calls 44
Obtaining A List Of Shared Resources 44
Summary Of Network System Calls 45
DOS Compatible Calls 45

tc "Calling Conventions" \l2§Calling Conventions

tc "Assembly Language" \l3§Assembly Language

Each NOS function is described using an entry in a standard format. For example:

5F80H

tc "Assembler Interface" \l3§Assembler Interface

INPUT:
AX
5F80H

BX
Login entry index (0 based)

ES:DI
Pointer to 16-byte buffer to receive logged in server name

OUTPUT:
NC
If no error

CY
If error occurred (or no more entries are available)

AX
Error code if error

BX
Same value used to call function (you must increment BX yourself to get next entry)

DL
Adapter number used for log in

ES:DI
Pointer to ASCIZ server name which does NOT include the \\ prefix

The **INPUT** section describes registers and values to use when calling the function. The **OUTPUT** section describes the information returned by the function. The outputs are always either in registers or in data areas addressed explicitly by an input parameter. Register identifiers are conventional Intel menmonics. Unless otherwise stated, NOS preserves registers across the call except for the flags. By convention, most function calls return the carry flag set to indicate an error (with the error number in AX), and leave the carry flag clear if the function was successful. Exceptions to this are noted in the description. References to ASCIZ strings mean null terminated strings of bytes.

tc "C Language" \l3§C Language

Each NOS function call supported by the C library (see Chapter 14) is specified like this:

extern int NOSGetLogin(int *npIndex, char *cpServer, int *npAdapter);

Pointer to int containing login entry index

Pointer to server name buffer

Pointer to int where returned adapter number is saved
You must include the **NOSLIB.H** header file in your program to obtain the necessary external declarations for all the NOS functions. The header file **NOS.H** contains declarations for all the data structures relevant to NOS programming.

tc "C Language Interface" \l2§Assembly Language Interface

The files that support the assembly language interface to NOS are as follows:

NOS.INC The include file that defines all NOS constants and data structures

NETBIOS.INC

tc "C Language Interface" \l2§C Language Interface

The files that support the C library interface to NOS are as follows:

NOS.H The header file that defines all NOS constants and data structures

NCB.H The header file that defines the NetBIOS NCB structures.

NOSLIB.H The header file that defines the function call interface to the C library

NOSLIB.LIB The object module library containing the NOS functions callable from C. (Note that versions of this file exist for both Microsoft C and Borland C.)

LIBTEST.C An API test program that includes an example of each NOS function call.

tc "The LANtastic Interface" \l2§The LANtastic Interface

The NOS services fall into two categories: standard DOS compatible network functions, and extended network functions specific to LANtastic. Since NOS is compatible with NetBIOS, the NetBIOS functions are also available using the INT 5C interface.

tc "Network Pathnames" \l3§Network Pathnames

Many functions refer to network paths. NOS network paths are fully compatible with DOS network paths. The general syntax for a network path is:

Where:

For example:

Refers to a file located on server **MAIN_SERVER**. The path required to access the file is **PROGS\DATA\SAMPLE.TXT**. The name **PROGS** is a shared resource name that refers to a network resource.

Some DOS networks (for example, the IBM-PC LAN) do not fully support paths for all systems calls (in particular **FIND FIRST**). NOS, however, supports full paths for all system calls that can take a path argument. This means that many public domain file and directory utilities will still operate when invoked with a network path.

tc "Error Codes And Error Handling" \l3§Error Codes And Error Handling

All DOS compatible system calls and all NOS specific system calls, return error codes. The error code number is always returned in AX. When an error occurs, the carry flag (CY) is set. The C library functions return a value of -1 and stores the error code in **NOSerrno**. NOS provides a mechanism for expanding the error code number into a text string.

This is a list of error codes that may be returned by NOS:

Error Number
Meaning

01H
Invalid function number

02H
File not found

03H
Path not found

04H
File open limit has been exceeded or no handles left

05H
Access denied

06H
Invalid handle

07H
Memory control blocks destroyed

08H
The memory limit has been exceeded

09H
Invalid memory block address

0AH
Invalid environment

0BH
Invalid format

0CH
Invalid access code

0DH
Invalid data

0EH
RESERVED

0FH
Invalid drive was specified

10H
Attempt to remove current directory

11H
Not same device

12H
No more files

13H
Attempt to write on write protected disk

14H
Unknown unit

15H
Drive not ready

16H
Unknown command

17H
Data CRC error

18H
Bad request structure length

19H
Seek error

1AH

Unknown media

1BH
Sector not found

1CH
No paper

1DH
Write fault

1EH
Read fault

1FH
General failure

20H
Sharing violation

21H
Lock violation

22H
Invalid disk change

23H
FCB unavailable

24H
Sharing buffer overflow

25H
RESERVED

26H
Cannot complete file operation

27H-31H
RESERVED

32H
Network request not supported

33H
Network node ?????????????? is not listening

34H
The name already exists on the network

35H
Cannot locate network name

36H
The network is busy

37H
Server connection to network node ?????????????? broken

38H
The NetBIOS command limit has been exceeded

39H
The network adapter has malfunctioned

3AH
Incorrect response received from network node ??????????????

3BH
Unexpected network error from network node ??????????????

3CH
Incompatible network node ??????????????

3DH
Print queue full on network node ??????????????

3EH
No room for print file on network node ??????????????

3FH
The print file has been deleted on network node ??????????????

40H
The network name has been deleted

41H
You have been denied access on network node ??????????????

42H
Invalid network device

43H
The network name was not found

44H
The network name limit has been exceeded

45H
The session limit has been exceeded

46H
Network node ?????????????? has been temporarily paused

47H
The network request to network node ?????????????? was denied

48H
Print or disk redirection is paused on network node ??????????????

49H
Invalid network version

4AH
Account has expired

4BH
Password has expired

4CH
Login attempt invalid at this time

4DH
Disk limit has been exceeded on network node ??????????????

4EH
Not logged into network node ??????????????

4FH
RESERVED

50H
The file already exists

51H
RESERVED

52H
Cannot make directory entry

53H
Failure on critical error

54H
Too many redirections or logins to network node ??????????????

55H
Duplicate redirection or login to network node ??????????????

56H
Invalid username or password

57H
Invalid parameter

58H
Network data fault

59H
Function not supported on network

5AH
Required system component not installed

When you expand the error code into a text string, NOS substitutes the network node name for the string of question marks in the error text. To expand an error code into a text string you need to issue a multiplex interrupt (INT 2FH) with the following parameters:

Assembler Interface

INPUT:
AH
5

AL
0 for installation check

for error code (in pre DOS 4.00)

1 or 2 for error code in DOS 4.00

BX
Error code

OUTPUT:
NC
If error code converted to text

CY
If error code can't be converted

ES:DI
Pointer to ASCIZ text buffer containing error text This is a read only text buffer and you must not alter the text in this buffer.

C Interface

nError
Error number

cpMessage
A *RESERVED* error code causes the error string to be set to "General failure". The maximum width for the question mark field is 15 characters. The error text is automatically adjusted to remove trailing spaces from the text buffer. For example the text returned for error 33H on node **HOST1** would be:

The following code sequence will work for DOS 3.x, 4.x and 5.x:

```
    ax, error_number
    ; Get error number
```

```
    bx, ax
    ; Place in BX also

    ah, 5

    2fh
```

Here is an example using the C library interface (the printf %Fs far string specifier is peculiar to Microsoft C):

```
    ...

    ...


    // Could not convert error number

    ...


    printf("Error %d: %Fs\n", NOSerrno, fpErrbuffer);
```

In addition, the routine **NOSperror()** has been defined for the NOS C library:

cpMessage
This routine uses the **NOSGetErrorText()** function to print the error message identified by the value of **NOSerrno**. Since **NOSerrno** is reset with every library call, you can use **NOSperror()** to print the most recent error message. **NOSperror()** prints the string parameter in the call, followed by a "**:** ", followed by the NOS error message. For example:

```
    NOSperror("NOSLogin");
```

Will print:

On standard output.

tc "Configuration Testing" \l2§Configuration Testing

tc "Testing For The Existence Of The NetBIOS" \l3§Testing For The Existence Of The NetBIOS

Before you can execute NetBIOS calls, you must install the NetBIOS software. On a LANtastic network you accomplish this by installing the Artisoft LANBIOS (**LANBIOS2** for 2Mbps networks, **AEX** and **AILANBIO** for Ethernet.) On a LANtastic network you can simply check for the presence of the LANtastic redirector. If the redirector is present then so is the NetBIOS.The following code fragment shows how to test directly for the presence of the NetBIOS.

```
; An illegal NCB is used to determine if the NetBIOS is present

illegal_NCB
db 7fh
; Illegal command

db 63 dup (0)
; Rest of NCB is 0
;
; NETBIOS_PRESENT - Determine if we have a NETBIOS present
;
; IN:
;
ES = Current Data segment
;
; OUT:
;
BX Destroyed
;
FLAGS
;
CARRY if not present

NETBIOS_present


es

ax, 355ch
; Get 5C vector
```

```
21H

ax, es

es

ax, 0F000h

no_vector
; Yes. Then not a real 5C

ax, bx
; Test for 0

have_vector

no_vector:

; Show not present




have_vector:

al, 0

bx, offset illegal_NCB

5cH

al, al
;Will get changed on illegal command

no_vector


NETBIOS_present
endp
```

tc "Testing For The Existence Of The Redirector, Server And " \ l3§Testing For The Existence Of The Redirector, Server And LANPUP

You can issue a multiplex interrupt (2FH) to determine if the redirector, server or LANPUP software are present. Note that if the redirector is present on a LANtastic network, then the LANBIOS must also be present. This is an easy way to test for NetBIOS presence.

Assembler Interface

INPUT:
AX
B800H

OUTPUT:
AL
0 If neither redirector or server installed

NZ
Redirector, server or LANPUP installed

BL
Contains bits indicating which software is installed

10000000b Redirector has pop up receive message capability.

01000000b Server is installed

00001000b Redirector is installed

00000010b LANPUP is installed

C Interface

extern int NOSPresence();

Example

```
// Error

if (i & 0x40)

printf("LANtastic server is installed");
```

tc "Testing For The Existence Of SHARE" \l3§Testing For The Existence Of SHARE

You can issue a multiplex interrupt (2FH) to determine if **SHARE** is present. Note that this function is officially an "undocumented" DOS function.

Assembler Interface

INPUT:
AX
1000H

OUTPUT:
AL
0FFh If SHARE is installed, 0 if not installed

C Interface

extern int NOSSharePresence();

tc "Determining The Network Software Version" \l3§Determining The Network Software Version

You can issue a multiplex interrupt (2FH) to determine which version of the network software is running.

Assembler Interface

INPUT:
AX
B809H

OUTPUT:
AH
Major version number

AL
Minor version number

C Interface

extern int NOSGetVersion();

Note
The version numbers are returned as decimal numbers. For example, version 3.03 would return AH equal to 3 and AL equal to 3. In C, this statement:

Prints:

Use this **printf** statement to print the correct version number:

Will print:

tc "Testing For The Alternate NetBIOS Interface - Interrupt 2AH" \ l3§Testing For The Alternate NetBIOS Interface - Interrupt 2AH

The normal interface to the NetBIOS is via interrupt 5CH. You can also call NetBIOS using the alternate interrupt 2AH. The LANtastic redirector supports the 2A interface as well. The LANtastic redirector intercepts 2A interrupts and reformats them to 5C interrupts. Here is a list of the supported interrupt 2A functions.

INT 2AH Function 0: INSTALLATION CHECK

Checks if an interrupt 2AH interface is installed.

Assembler Interface

INPUTS:
AH
0

OUTPUTS:
AH
0 if not installed

not 0 if installed

C Interface

extern int NOSCheck2A();

INT 2AH Function 1: EXECUTE NetBIOS WITH NO ERROR RETRY

Executes a NetBIOS command.

Assembler Interface

INPUTS:
AX
01xxH or 0401H to execute NetBIOS with no error retry

0400H to execute NetBIOS with error retry

Error codes that are automatically retried are:

No sessions resources (09H)

No listen (12H)

Interface busy (21H)

ES:BX
Pointer to NCB

OUTPUTS:
AL
NetBIOS error code

AH
0 if no error

1 if error

C Interface

extern int NOSExecNetBIOS(BOOL bRetry, struct Ncb *cpNcb);

TRUE to execute NetBIOS with error retry, FALSE otherwise

Points to NetBIOS NCB

INT 2AH Function 5: GET RESOURCE INFORMATION

Returns NetBIOS resources that are available for use.

Assembler Interface

INPUTS:
AH
5

OUTPUTS:
BX
Available names (16 - names in use)

CX

Available NCBs (free NCBs)

DX
Available sessions (max sessions minus pending sessions)

C Interface

extern int NOSGetResources(int *npNames, int *npNcbs, int *npSessions);

Pointer to int where # of available names will be stored

Pointer to int where # of available NCBs will be stored

Pointer to int where # of available sessions will be stored

Example

```
// Error

printf("%d names, %d NCBs, %d sessions", nNames, nNcbs, nSessions);
```

tc "Controlling Redirected Printer Streams" \l2§Controlling Redirected Printer Streams

You can control how LANtastic sends output to redirected printers in three ways:

- Œ You can set a combine mode that does not separate multiple print jobs when programs terminate or when the printer is opened.
- Œ You can set a separate mode that separates print jobs when a program terminates or when the printer is opened and closed.
- Œ You can flush redirected output, thereby forcing the printer to begin printing.

These functions implement the **NET** program's **NET LPT COMBINE**, **NET LPT SEPARATE** and **NET LPT FLUSH** commands. Two software interfaces allow you to control these functions. These interfaces consist of the interrupt 2AH interface and a standard interrupt 21H interface.

Note

A bug in LANtastic version 4.0 causes the interrupt 2A printer control functions to have no effect. The interrupt 21 functions work correctly.

Note

The DOS **COMMAND.COM** program always issues a "flush spooled output" call when it prompts for command input. This has the effect of forcing print jobs to be separated.Therefore these interrupts are ineffective across multiple program invocations performed through **COMMAND.COM**. The only exception to this is within DOS batch files, since **COMMAND.COM** does not prompt for command input while executing a batch file.

tc "Controlling Printers With Interrupt 2A" \l3§Controlling Printers With Interrupt 2A Functions

SET SPOOLED OUTPUT TO COMBINE MODE

All printer output is combined into a single print job regardless of the printer being opened, closed or programs terminating. Flush printer output calls will still be honoured however, and separate print jobs will be spooled.

Assembler Interface

INPUTS:
AX
0601H

OUTPUTS:
None

C Interface

extern int NOSSetCombine();

SET SPOOLED OUTPUT IN SEPARATE MODE

Printer output is not combined when multiple programs are run, or when the printer is opened or closed. This command implicitly starts a new print job.

Assembler Interface

INPUTS:
AX
0602H

OUTPUTS:
None

C Interface

extern int NOSSetSeparate();

FLUSH PRINTER OUTPUT

Printer output is flushed and a new print job is started. If no output exists to be flushed, then this function has no effect.

Assembler Interface

INPUTS:
AX
0603H

OUTPUTS:
none

C Interface

extern int NOS2AFlush();

tc "Controlling Printers With Interrupt 21" \l3§Controlling Printers With Interrupt 21 Functions

RETURN REDIRECTED PRINTER MODE

Returns the current printer mode.

Assembler Interface

INPUTS:
AX
5D07H

OUTPUTS:
DL
0 Redirected output is being combined

1 Redirected output is being separated

C Interface

extern int NOSGetPrinterMode();

Returns the value of DL as the function value: 0 means combined, 1 means separated

SET REDIRECTED PRINTER MODE

Sets the current printer mode. Setting separated mode implicitly starts a new print job.

Assembler Interface

INPUTS:
AX
5D08H

DL
0 Set redirected output to be combined
1 Set redirected output to be separated.

OUTPUTS:
None

C Interface

extern int NOSSetPrinterMode(int nMode);

0 means combine, 1 means separate

FLUSH PRINTER OUTPUT

Printer output is flushed and a new print job is started. If no output exists to be flushed, then this function has no effect.

Assembler Interface

INPUTS:
AX

5D09H

OUTPUTS:
None

C Interface

extern int NOSFlush();

tc "DOS Compatible Network Functions" \l2§DOS Compatible Network Functions

LANtastic supports the standard DOS network system call functions with the exception of GET and SET PRINTER SETUP STRING (5E02H and 5E03H). Although you can issue these calls from a program they have no effect. This is because LANtastic relies upon the system manager to define the printer setup strings using the **NET_MGR** program.

GET MACHINE NAME

GET MACHINE NAME returns the name that your machine is known by on the network. You may use the name and NetBIOS name number returned by this function to perform NetBIOS commands. You must not use any other NetBIOS names added by LANtastic .

Assembler Interface

INPUT:
AX
5E00H

DS:DX
Pointer to 16 byte buffer where ASCIZ machine name is returned

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

CL

NetBIOS name number of machine name on first adapter used by the redirector.

CH
0 If machine name is not set

Not zero if machine name has been set

DS:DX
Points to ASCIZ machine name

Note

On return the machine name is space padded up to the full fifteen characters allowed in the name. The last byte is set to null. All fifteen characters are significant for NetBIOS names.

C Interface

extern int NOSGetMachineName(char *cpName, int *npName, BOOL *pbName);

Points to buffer to hold machine name

Points to int to save NetBIOS name number

Points to BOOL, set TRUE if name is set, FALSE if not

Example

```
char cpName[16];
int i;
int nNBname;
BOOL bNameset;

if (NOSGetMachineName(cpName, &nNBName, &bNameset) == -1)

else {


printf("Machine name not set");


// Null terminate the name

for (i = 0; i < 16; i++)

if (cpName[i] == ' '){

cpName[i] = '\0';

break;

}

printf("Machine is %s, NetBIOS name # is %d", cpName, nNBname);
```

SET PRINTER SETUP

This function initializes a setup string that is sent to the network printer at the start of each print job.

Assembler Interface

INPUT:
AX
5E02H

BX
Redirection list index (0 based)

CX
Setup string size

DS:SI
Pointer to printer setup string

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSSetPrinterSetup(int nIndex, int nLen, char *cpStr);

Redirection list index

Length of setup string (in case it contains nulls)

Pointer to setup string

Note

LANtastic does not process this request since printer setup strings are controlled by the system administrator using the **NET_MGR** program. The system call does *not* return an error. The C library faithfully calls NOS despite the fact that nothing happens.

GET PRINTER SETUP

GET PRINTER SETUP will return the setup string set with the SET PRINTER SETUP function.

Assembler Interface

INPUT:
AX
5E03H

BX
Redirection entry index (0 based)

ES:DI
Pointer to buffer to contain setup string

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

CX
Length of setup string

C Interface
extern int NOSGetPrinterSetup(int nIndex, int *npLen, char *cpStr);

Redirection list index

Points to int where length of setup string is saved

Pointer to buffer for setup string

Note

LANtastic does not process this request since printer setup strings are controlled by the system administrator using the **NET_MGR** program. The system call does *not* return an error, however, and the length of the setup string (CX) returned will be 0. Again, the C library calls NOS despite this lack of activity.

GET REDIRECTED DEVICE ENTRY

GET REDIRECTED DEVICE ENTRY returns information about a single redirected device (see REDIRECT DEVICE function.) You can use this function to build a list of redirected devices.

Assembler Interface

INPUT:
AX
5F02H

BX
Redirection entry index (index 0 specifies the first entry)

DS:SI
Pointer to 16-byte buffer for local device name

ES:DI
Pointer to 128-byte buffer to receive the network path referred to by the redirected device

OUTPUT:
NC

If no error

CY
If error occurred

AX
Error code if error

BL
Type of device: 03H for printer, 04H for disk

CX
Value stored by REDIRECT DEVICE function (should always be 0 for NOS)

DX
May be destroyed

BP
May be destroyed

DS:SI
ASCIZ device name

ES:DI
ASCIZ network path

C Interface

extern int NOSGetRedirDevice(int nIndex, char *cpName, char *cpPath,

WORD *wpDevice);

int holding redirected device index

Points to buffer for device name

Points to buffer for path

Points to WORD to hold device type. 0xFF masks the device type.

Example

```
printf("Valid device, type %d", wFlags & 0xFF);
```

REDIRECT DEVICE

REDIRECT DEVICE allows you to make network connections between local devices and network paths.

Assembler Interface

INPUT:
AX
5F03H

BL
Type of device, 03H Printer, 04H Disk

CX
Value to store with redirected device (Set this to 0 for compatibility with NOS)

DS:SI
Pointer to ASCIZ local device name. For printer devices specify: PRN, LPT1, LPT2, LPT3, COM1 or COM2. For disk devices specify A:, B:, C:, ...

ES:DI
Pointer to network path for redirected device

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSRedirDevice(int nDevice, char *cpName, char *cpPath);

Device type

(Note that no CX equivalent is required, it's always 0)

Points to name of device

Points to network path
Example

CANCEL DEVICE REDIRECTION

CANCEL DEVICE REDIRECTION allows you to break a network connection and restore the local device to its former state.

Assembler Interface

INPUT:
AX
5F04H

DS:SI
Pointer to ASCIZ device name

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSCancelRedir(char *cpName);

Points to device name

tc "NOS Extended Network Functions" \l2§NOS Extended Network Functions

NOS provides an extended set of network system calls that allow you access to all the LANtastic functions. These system calls are significantly extended beyond the few calls defined by DOS.

Several functions retrieve indexed data. You always pass the index parameter in the **BX** register. The index is always zero based unless otherwise noted. Upon return from the function call, **BX** will contain the index of the *next* table entry. The index of the current entry will be **BX** minus 1. If the index passed to the function call in **BX** refers to an invalid entry, NOS will increment **BX** until it finds a valid entry. The call returns this entry and **BX** is incremented. In these cases, you must assume only that an index value of **BX** minus 1 is valid. Some functions take such an index as a parameter, but do *not* increment it as described. Study the description of each function call carefully to see whether the call changes the index.

GET LOGIN ENTRY

Assembler Interface

INPUT:
AX
5F80H

BX
Login entry index (0 based)

ES:DI
Pointer to 16-byte buffer to receive logged in server name

OUTPUT:
NC
If no error

CY
If error occurred (or no more entries are available)

AX
Error code if error

BX
Same value used to call function (you must increment BX yourself to get the next entry)

DL
Adapter number used for log in

ES:DI
Pointer to ASCIZ server name which does *not* include the \\ prefix

C Interface

extern int NOSGetLogin(int *npIndex, char *cpServer, int *npAdapter);

Pointer to int containing login entry index

Pointer to server name buffer

Pointer to int where returned adapter number is saved

Example

```
// Add double slash prefix


printf("Entry %d, logged into %s on adapter %d\n",

nIx, cpServer, nA);

nIx++;
```

Note The login entry index (the BX input parameter) corresponds to the index used in the GET USERNAME ENTRY call (5F83h). A combination of these calls allows you to enumerate current logged in users.

LOGIN TO A SERVER

Assembler Interface

INPUT:
AX
5F81H

ES:DI
Pointer to server, username and password in the form \\server\username<0>password<0>. (For LANtastic version 3.0, if no password is used, the string must be: \\server\username<0><0><0>. In LANtastic version 4.0, two null characters are acceptable.)

BL
Adapter number to use for log in attempt, 0FFH to try all valid adapters, 0-7 to use an adapter explicitly.

OUTPUT:
FLAGS
NC If no error

CY If error occurred

AX
Error code if error

C Interface

extern int NOSLogin(char *cpLogin, int nAdapter);

Pointer to server, username and password

Adapter number to use

Note

LANtastic will convert the input string to uppercase before use.

LOGOUT OF A SERVER

Assembler Interface

INPUT:
AX
5F82H

ES:DI
Pointer to ASCIZ server name in the form \\server

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSLogout(char *cpServer);

Pointer to server name

GET USERNAME ENTRY

Assembler Interface

INPUT:
AX
5F83H

BX
Username entry index (0 based). Will not be updated.

ES:DI
Pointer to 16-byte buffer to receive username used for this login

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

BX
Username entry index (not updated.)

DL
Adapter number used for log in

ES:DI
Pointer to ASCIZ username

C Interface

extern int NOSGetUserName(int *npIndex, char *cpName, int *npAdapter);

Pointer to int holding username entry index (not updated)

Pointer to buffer for username

Pointer to int to receive adapter number
Example

printf("Entry %d, user %s on adapter %d\n",

nIx, cpUser, nA);

nIx++;

Note
The login entry index (the BX input parameter) corresponds to the index used in the GET LOGIN ENTRY call (5F80h). A combination of these calls allows you to enumerate current logged in users.

GET INACTIVE SERVER ENTRY

Assembler Interface

INPUT:
AX
5F84H

BX
Non-logged in server index

ES:DI
Pointer to 16-byte buffer to receive a server name that you are *not* logged in to, but that is available for logging in

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

BX
Same value used to call function (you must increment BX yourself to get the next entry)

DL
Adapter number the inactive server is on (this number may be used as input to the LOGIN TO SERVER (5F81H, NOSLogin()) function)

ES:DI
Pointer to ASCIZ server name which does *not* include the \\ prefix

C Interface

extern int NOSGetServer(int *npIndex, char *cpServer, int *npAdapter);

Pointer to int holding server name entry index

(This is *not* updated.)

Pointer to buffer for server name

Pointer to int to receive adapter number
Example

```
printf("Server %s available on adapter %d\n",

cpServer, nA);

nIx++;
```

Note

The redirector maintains the list of available servers. The number of entries in the list is determined by the value of the LOGINS= switch on the redirector command line. On a network with more servers present, only the most recently active server names will be stored. (For example, servers that you recently logged out from.) Servers with their "Send Server ID" capability disabled will never appear in this list.

CHANGE PASSWORD

Assembler Interface

INPUT:
AX
5F85H

ES:DI
Pointer to string in the form:
\\server\old-password<0>new-password<0>
You must be logged into the server.

This operation is illegal for group accounts.

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSChangePassword(char *cpStr);

Pointer to buffer holding a string of the form:

\\server\old-password\0\new-password\0

DISABLE ACCOUNT

DISABLE ACCOUNT disables the current logged in account. It applies only when concurrent login entries are set to 1 (by **NET_MGR**.) The account must then be reenabled by the system manager. After a DISABLE ACCOUNT operation, a logged in user may continue to use the account until they log out.

Assembler Interface

INPUT:
AX
5F86H

ES:DI
Pointer to ASCIZ server and password string in the form \\server\password. You must be logged into the server.

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSDisable(char *cpStr);

Pointer to \\server\password string

GET ACCOUNT

GET ACCOUNT returns the information for the account used to log into this server.

Assembler Interface

INPUT:
AX
5F87H

DS:SI
Pointer to 128-byte buffer to receive account information

ES:DI
Pointer to ASCIZ server string in the form \\server. You must be logged into the server

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

BX
Destroyed (though not used as an input parameter)

C Interface

```
extern int NOSGetAccount(struct user_account *cpAccount, char *cpServer);
```

Pointer to account structure

Pointer to server name
The buffer format is as follows:

```
user_account
UA_name              db  16 dup (?)  ; Zero padded username
UA_internal          db  16 dup (0)
UA_description       db  32 dup (?)  ; Full user description
UA_privilege         db  ?           ; Privilege bits (see bit mask
                                     ; definitions below)
UA_concurrent        db  ?           ; Maximum concurrent log ins
UA_allowed_times     db  42 dup (?)  ; 1 bit for each half hour for 7 day
                                     ; week beginning on Sunday.
                                     ; 0 means allowed.
UA_internal2         dw  ?
UA_last_login_time   dw  2 dup (?)   ; Last time logged in
UA_account_expiration  dw  2 dup (?)   ; Expiration date (DOS-format)
                                     ; Year, Month:Day
UA_password_expiration dw  2 dup (?)   ; Expiration date (as above),
                                     ; 0 Means no expiration date
UA_password_extension  db  ?         ; 1-31  Number of days to reextend
                                     ; password after change
                                     ; 0 No extension required
UA_undelete_char     db  ?           ; First letter of UA_name when
                                     ; account is deleted (first
                                     ; character of UA_name is replaced
                                     ; with a zero.)
UA_xprivilege        db  ?           ; Extended privilege

UA_future            db  128 - UA_future dup (?)
user_account
```

Privilege bits for UA_privilege:

```
UA_privilege_superACL   equ 10000000b ; Bypass ACLs
UA_privilege_superqueue equ 01000000b ; Bypass queue protection
UA_privilege_peer       equ 00100000b ; Treat as local process
UA_privilege_supermail  equ 00010000b ; Bypass mail protection
```

```
UA_privilege_audit      equ 00001000b ; The user can create audit entries
UA_privilege_system     equ 00000100b ; The user has system manager privileges
UA_xprivilege_nopwchange equ 00000001b; User cannot change password
```

Example

```
printf("%s logged into %s", U.UA_name, cpServer);
```

LOGOUT FROM ALL SERVERS

This call is equivalent to performing a logout call for all the servers that you are currently logged into.

Assembler Interface

INPUT:
AX
5F88H

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

```
extern int NOSLogoutAll();
```

COPY FILE

COPY FILE copies the source file (designated by the source handle) to the destination file (designated by the destination handle.) The copy is performed by the server and requires no workstation resources. Both files must reside on the same server.

Assembler Interface

INPUT:
AX
5F97H

CX:DX
Amount to copy (set to FFFF:FFFF to copy entire file)

SI
Source handle

DI
Destination handle

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

DX:AX
Amount copied if successful

C Interface

extern int NOSCopyFile(DWORD *dwAmount, int nSource, int nDest);

Pointer to long containing amount to be copied.

(Updated with actual amount copied.)

Source file handle

Destination file handle

Example

```
// Error processing and header files not shown here



printf("Copied %ld bytes", amount);
```

SEND UNSOLICITED MESSAGE

This function uses the NOS message service to transmit messages across the network. The messages have a fixed format. Transmission of a message to yourself is perfectly legal (and useful.)

This is the format of a message buffer. The 16-byte name fields (**MB_machine**, **MB_server** and **MB_user**) are null terminated and contain the characters to be matched. For example AB<0> will match ABxxxxxx. Note that a null string will match any other string, that is, any network name and will send the message to all machines on the network. The server names should not contain leading '\\' characters. The names are case sensitive. You should convert any strings to be uppercase only to match LANtastic conventions.

You can use the **MB_server** and **MB_user** fields to restrict the message recipients. For example, setting the **MB_machine** and **MB_user** fields to the null string and the **MB_server** field to "XSERVER" will send the message to all users currently logged into the machine XSERVER.

```
message_buffer
struc
```

```
MB_reserved    db  ?          ; Reserved field used by system call
MB_type        db  ?          ; User defined message type (see bit mask
                              ; definitions below)
MB_machine     db  16 dup (?) ; Machine name destination
MB_server      db  16 dup (?) ; User must be logged into this server
MB_user        db  16 dup (?) ; User must be using this username
MB_originator  db  16 dup (?) ; Originator's machine name. Filled in when
                              ; message is received
MB_text        db  80 dup (?) ; Message text
message_buffer
```

Message buffer type. The MBT_general type is used by **NET** and **LANPUP** to send messages.

```
MBT_general    equ  0     ; General message used by NET, LANPUP and others
MBT_warning    equ  1     ; Server warning message
```

Assembler Interface

INPUT:
AX
5F98H

DS:SI
Pointer to message buffer

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSSendMsg(struct message_buffer *cpMsg);

Pointer to message buffer

Note

Currently, no errors are returned by NOS.

GET LAST RECEIVED UNSOLICITED MESSAGE

This function allows you to retrieve a copy of the last message received on this machine.

Assembler Interface

INPUT:
AX
5F99H

ES:DI
Pointer to message buffer (see function 5F98H for buffer format)

OUTPUT:
NC
If no error

CY
If error occurred (or if no message is pending)

AX
Error code if error

C Interface

extern int NOSGetMsg(struct message_buffer far *cpMsg);

Address of message buffer to copy message to.

GET MESSAGE PROCESSING FLAG

This function returns a flag indicating the current disposition of the NOS message service on this machine.

Assembler Interface

INPUT:
AX
5F9AH

OUTPUT:
DL
Flag describing what processing should be done when an unsolicited message is received

NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSGetMsgFlag(int *npFlag);

Pointer to int to receive copy of message processing flag

Format of returned message processing flag:

```
MPB_beep        equ 00000001b  ; Beep before message delivered
```

```
MPB_deliver     equ 00000010b  ; Deliver message to message service
MPB_auto_pop_up equ 00000100b  ; Pop up messages automatically
```

SET MESSAGE PROCESSING FLAG

This function allows you to control the disposition of the LANtastic message service.

Assembler Interface

INPUT:
AX
5F9BH

DL
Bits describing what processing should be done when an unsolicited message is received (see GET MESSAGE PROCESSING FLAG function for the format)

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSSetMsgFlag(int nFlag);

New value for message processing flag

POP UP LAST RECEIVED MESSAGE

POP UP LAST RECEIVED MESSAGE uses the LANtastic message service to display the last received message on the screen. The message is placed on the specified line for the specified time. When LANtastic removes the message, the original screen contents are restored. The user can remove the message before the display time elapses by pressing the **Esc** key. LANtastic cannot pop up a message if the screen is in a graphics mode.

Assembler Interface

INPUT:
AX
5F9CH

CX
Amount of time, in ticks, to leave message on the screen (there are 18.2 ticks per second)

DH
Line number where message will be placed (top line is 0, no validity checking is done)

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSPopUpMsg(int nTicks, int nLine);

Number of ticks

Line number

Note

The only error currently returned is 0BH (Invalid format). NOS returns this error if the screen is not currently in a text mode and the message cannot be displayed.

GET QUEUE ENTRY

Assembler Interface

INPUT:
AX
5FA0H

BX
Queue entry index (0 for first entry)

DS:SI
Pointer to 162-byte buffer to receive queue entry information

ES:DI
Pointer to ASCIZ server in the form \\server

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

BX
Next queue entry index

DS:SI
Filled queue entry information buffer

C Interface

extern int NOSGetQueue(int *npIndex, struct queue_entry *cpQueue,

char *cpServer);

Pointer to int holding queue entry index

(This *will be* updated.)

Pointer to buffer to hold queue entry information

Pointer to buffer for server name
Definition of a queue entry:

```
queue_entry       struc
QE_status         db   ?         ; Status of queue entry
QE_size           dd   ?         ; Size of spooled file
QE_type           db   ?         ; Type of queue entry
QE_output_control  db   ?        ; Control of despooled file
QE_copies         dw   1         ; Number of copies
QE_sequence       dd   ?         ; Sequence number of queue entry
QE_spooled_file   db   48 dup (?) ; Pathname of spooled file
QE_user           db   16 dup (?) ; Username who spooled file
QE_machine        db   16 dup (?) ; Machine name user was on
QE_date           dw   ?         ; Date file spooled (DOS format)
QE_time           dw   ?         ; Time file spooled (DOS format)
QE_destination    db   17 dup (?) ; ASCIZ Device name or username destination
QE_comment        db   48 dup (?) ; Comment field
queue_entry       ends
```

Different queue entry statuses:

```
QE_status_free       equ   0  ; The queue entry is empty
QE_status_update     equ   1  ; The queue entry is being updated
QE_status_hold       equ   2  ; The queue entry is held
QE_status_wait       equ   3  ; The queue entry is waiting for despool
QE_status_active     equ   4  ; The queue entry is being despooled
QE_status_cancel     equ   5  ; The queue has been canceled
```

```
QE_status_file_error  equ   6  ; The spooled file could not be accessed
QE_status_spool_error equ   7  ; The destination could not be accessed
QE_status_rush        equ   8  ; Rush this job
```

Different types of queue entries:

```
QE_type_print        equ   0  ; Spooled printer queue file
QE_type_message      equ   1  ; Spooled message (mail)
QE_type_local_file   equ   2  ; Spooled local file
QE_type_remote_file  equ   3  ; Spooled remote file
QE_type_modem        equ   4  ; Spooled to remote modem
QE_type_batch        equ   5  ; Spooled batch processor file
```

Definitions for output control settings:

```
QE_OC_keep   equ 01000000b  ; Keep after despooling (don't delete)
                            ; For mail - allow delete only by owner
QE_OC_voice  equ 00100000b  ; For mail - mail file contains voice data
QE_OC_opened equ 00010000b  ; For mail - message has been read
QE_OC_request_response equ 00001000b

 ; For mail, a response is requested
```

SET QUEUE ENTRY

SET QUEUE ENTRY allows you to set certain fields in newly created queue entry. You create a queue entry by performing an open or create call on the file **\\server\@MAIL** (for mail messages) or **\\server\@resource** (for printer queue entries.) You then use the returned handle in the SET QUEUE ENTRY call. You cannot perform a SET QUEUE ENTRY on an existing queue entry by opening the file **\\server\spool_filename**. The only way to match a queue entry returned by the GET QUEUE ENTRY call, and the current open queue entry is to manually compare the individual field settings with expected values. The only fields that can be altered by a SET QUEUE ENTRY call are:

(only for **@MAIL** queue entries)

Assembler Interface

INPUT:
AX

5FA1H

BX
Handle of queue entry (see notes above.)

DS:SI
Pointer to queue information buffer (see function 5FA0 for buffer format)

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSSetQueue(int *npHandle, struct queue_entry *cpQueue);

Pointer to int holding queue entry index (*not* updated)

Pointer to buffer holding queue entry information

CONTROL QUEUE

CONTROL QUEUE allows you to manipulate print jobs and physical printer despooling. You must have the proper privilege (**Q**) to execute the commands marked with an "*****".

```
CQ_start     equ   0   ;*Start despooling
CQ_halt      equ   1   ;*Halt despooling
CQ_halt_EOJ  equ   2   ;*Halt despooling at end of job
CQ_pause     equ   3   ;*Pause the despooler at end of job
CQ_single    equ   4   ;*Print single job
CQ_restart   equ   5   ;*Restart the current queue entry
CQ_cancel    equ   6   ; Cancel the current queue entry
CQ_hold      equ   7   ; Hold the queue entry
CQ_release   equ   8   ; Release a held queue entry
CQ_rush      equ   9   ;*Make the queue entry a rushed job
```

Assembler Interface

INPUT:
AX
5FA2H

BL
Queue control command (commands listed above)

ES:DI
Pointer to ASCIZ server in the form \\server<0>

For Cancel, Hold, Release and Rush commands:

CX:DX
Queue sequence number to control

For Start, Halt, Halt_EOJ, Pause, Single and Restart commands:

DX
Physical printer number:

0, 1, 2 for **LPT1**, **LPT2**, **LPT3**; 3, 4 for **COM1**, **COM2**. Any other value means all printers

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSControlQueue(int *npPtr, int nCmd, char *cpServer,

DWORD dwSeq);

Pointer to int holding printer number

Queue control command

Pointer to server name

Queue sequence number

GET PRINTER STATUS

Assembler Interface

INPUT:
AX
5FA3H

BX
Physical printer number: 0, 1, 2 for **LPT1**, **LPT2**, **LPT3**; 3, 4 for **COM1**, **COM2**. All other values mean all printers

DS:SI
Pointer to 15-byte buffer to receive printer status information (struct PS)

ES:DI
Pointer to ASCIZ server in the form \\server<0>. You must be logged into the server.

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

BX
Next physical printer number

DS:SI
Filled printer status buffer

C Interface

extern int NOSGetStatus(int *npPtr, struct PS *cpPs, char *cpServer);

Pointer to printer number (will be updated)

Pointer to printer status buffer

Pointer to server name
If the printer is actively printing, then more detailed information is also returned.

Definition of a printer status entry:

```
PS              struc
PS_state        db  ?   ; Printer state (defined below)
PS_index        dw  ?   ; Queue index corresponding to print job being
                        ; despooled. (-1 if not despooling:  ignore rest
                        ; of fields)
PS_CPS          dw  ?   ; Actual characters per second being output
PS_output_chars dd  ?   ; Characters actually output so far
PS_read_chars   dd  ?   ; Characters actually read from despooled file so
                        ; far. May be used to compute percent completed.
PS_copies       dw  ?   ; Copies remaining to print
PS              ends
```

Definition of printer states:

```
printer_state       record  PS_state_pause:1, PS_state_value:7
PS_state_disabled   equ 0   ; Printer is disabled
PS_state_single_job equ 1   ; Printer will stop at end of job
PS_state_multijob   equ 2   ; Printer should print multiple jobs
```

GET STREAM INFO

Assembler Interface

INPUT:
AX
5FA4H

BX
Stream index number (0 based)

DS:SI
Pointer to 13-byte buffer to receive stream information

ES:DI
Pointer to ASCIZ server in the form \\server<0>. You must be logged into the server.

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

BX
Next stream number

C Interface

extern int NOSGetStreamInfo(int *npIndex, struct logical_stream *cpStream,

char *cpServer);

Pointer to int holding stream index (*will* be updated)

Pointer to stream buffer

Pointer to buffer containing server name
Each stream contains a logical printer resource template and a flag that indicates if jobs printed for that logical printer resource should be queued or not.

Definition of logical stream entry:

```
logical_stream struc
LS_queue       db  ?         ; 0 Disabled, non-zero Enabled
LS_template    db  12 dup (?) ; Template may contain ?'s (include "." as in
                              ; @???????.???)
logical_stream ends
```

SET STREAM INFO

Assembler Interface

INPUT:
AX
5FA5H

BX
Stream index number (0 based)

DS:SI
Pointer to 13-byte buffer which contains stream information (see function 5FA4H for buffer format)

ES:DI
Pointer to ASCIZ server in the form \\server<0>. You must be logged into the server.

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSSetStreamInfo(int *npIndex, struct logical_stream *cpStream,

char *cpServer);

Pointer to int holding stream index

Pointer to stream buffer

Pointer to buffer containing server name

CREATE USER AUDIT ENTRY

This function can be used to perform the **NET AUDIT** command. You must have the **U** privilege to use it successfully.

Assembler Interface

INPUT:
AX
5FA7H

DS:DX
Pointer to 8-byte (max) ASCIZ reason code

DS:SI
Pointer to 128-byte ASCIZ variable reason string

ES:DI
Pointer to ASCIZ server in the form \\server<0>. You must be logged into the server.

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSCreateAudit(char *cpCode, char *cpReason, char *cpServer);

Pointer to reason code

Pointer to reason string

Pointer to buffer holding server name

GET ACTIVE USER INFORMATION

Assembler Interface

INPUT:
AX
5FB0H

BX
Login entry index of server (0 based)

DS:SI
Pointer to 44-byte buffer to receive a server login entry

ES:DI
Pointer to ASCIZ server in the form \\server<0>

OUTPUT:
NC
If no error

CY

If error occurred

AX
Error code if error

BX
Next login entry index

DS:SI
Filled buffer with login entry information (buffer format is described below)

C Interface

extern int NOSGetUserInfo(int *npIndex, struct active_user_entry *cpInfo,

char *cpServer);

Pointer to int holding login entry index (updated)

Pointer to buffer to hold login entry

Pointer to buffer for server name
Definition of an active user entry:

```
active_user_entry
struc
AUE_VCID        dw   0          ; Virtual circuit number
AUE_state       db   ?          ; Login state (defined below)
AUE_command     db   ?          ; Last command issued
AUE_IO          db   5 dup (?)  ; Number of I/O bytes (40 bit number)
AUE_requests    db   3 dup (?)  ; Number of server requests (24 bit number)
AUE_name        db   16 dup (?) ; Name of logged in user
AUE_machine     db   16 dup (?) ; Name of remote logged in machine
AUE_xprivilege  db   ?          ; Extended privileges
AUE_time_left   dw   ?          ; Time left in minutes (0 is unlimited)
active_user_entry
ends
```

Definition of the various login states and privileges:

```
AUE_state_starting      equ  00000000b ; We are in the middle of a login
AUE_state_in            equ  00000001b ; We are fully logged in
AUE_state_RPL           equ  00000010b ; Remote program load login


AUE_privilege_superACL   equ  10000000b ; Bypass ACLs
AUE_privilege_superqueue equ  01000000b ; Bypass queue protection
AUE_privilege_peer       equ  00100000b ; Treat as local process
AUE_privilege_supermail  equ  00010000b ; Bypass mail protection
AUE_privilege_audit      equ  00001000b ; The user can create audit entries
AUE_privilege_system     equ  00000100b ; The user has system manager
                                        ; privileges
AUE_xprivilege_nopwchange equ 000000001b; User cannot change his password
```

Active user entry commands:

```
AUEC_login          equ  0    ; Login into a server
AUEC_terminate      equ  1    ; Process termination
AUEC_open           equ  2    ; Open a file
AUEC_close          equ  3    ; Close a file
AUEC_create         equ  4    ; Create a file if it's there or not
AUEC_new            equ  5    ; Create a new file that is not there
AUEC_unique         equ  6    ; Create a unique file
AUEC_commit         equ  7    ; Commit disk data to disk
AUEC_read           equ  8    ; Read from file
AUEC_write          equ  9    ; Write to file
AUEC_delete         equ  10   ; Delete file
AUEC_set_attr       equ  11   ; Set file attributes
AUEC_lock           equ  12   ; Lock byte range
AUEC_unlock         equ  13   ; Unlock byte range
AUEC_create_dir     equ  14   ; Create a subdirectory
AUEC_delete_dir     equ  15   ; Delete a subdirectory
AUEC_rename_file    equ  16   ; Rename a file
AUEC_find_first     equ  17   ; Find first matching file
AUEC_find_next      equ  18   ; Find the next matching file
AUEC_disk_free      equ  19   ; Get disk free space
AUEC_get_queue      equ  20   ; Get a queue entry
AUEC_set_queue      equ  21   ; Set a queue entry
AUEC_control_queue  equ  22   ; Control the queue
AUEC_get_login      equ  23   ; Return login information
AUEC_get_link       equ  24   ; Return link description
AUEC_seek           equ  25   ; Seek to a file position
AUEC_get_time       equ  26   ; Get server's time
AUEC_audit          equ  27   ; Create audit entry
AUEC_multi_open     equ  28   ; Open file in a multitude of modes
```

```
AUEC_change_password equ  29   ; Change a password
AUEC_disable_account equ  30   ; Disable account from further log ins
AUEC_copy_file       equ  31   ; Local server file copy
AUEC_get_username    equ  32   ; Get a username from account file
AUEC_translate_path  equ  33   ; Translate a server's logical path
AUEC_create_indirect equ  34   ; Make indirect file
AUEC_get_indirect    equ  35   ; Get indirect file text
AUEC_printer_status  equ  36   ; Printer status obtained
AUEC_get_stream      equ  37   ; Get logical print stream information
AUEC_set_stream      equ  38   ; Set logical print stream information
AUEC_get_account     equ  39   ; Get an account record
AUEC_shutdown        equ  40   ; Request server shutdown
AUEC_cancel_shutdown equ  41   ; Cancel server shutdown
AUEC_stuff           equ  42   ; Stuff server's keyboard
AUEC_write_with_commit equ 43  ; Write then commit data to disk
```

Example

```
printf("User %s logged in from %s\n",

U.AUE_name, U.AUE_machine);
```

GET SHARED DIRECTORY INFORMATION

Assembler Interface

INPUT:
AX
5FB1H

DS:SI
Pointer to 64-byte buffer to receive shared resource description

ES:DI
Pointer to ASCIZ server and resource in the form \\server\shared-resource

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

CX
ACL privilege bits for requesting user

DS:SI
ASCIZ description of shared resource

C Interface
extern int NOSGetDirInfo(char *cpRes, char *cpServer, int *npAcl);

Pointer to 64 byte buffer to hold resource definition

(This is returned as an ASCIZ string)

Pointer to buffer holding server name and resource

Pointer to int to hold ACL bit settings

ACL bit definitions:

```
ACL_read        equ 1000000000000000b  ; (R) Allow open for read and reading
ACL_write       equ 0100000000000000b  ; (W) Allow open for write and writing
ACL_create_file equ 0010000000000000b  ; (C) Allow file creation
```

```
ACL_create_dir  equ 0001000000000000b  ; (M) Allow directory creation
ACL_lookup      equ 0000100000000000b  ; (L) Allow file/directory lookups
ACL_delete_file equ 0000010000000000b  ; (D) Allow file deletion
ACL_delete_dir  equ 0000001000000000b  ; (K) Allow directory deletion
ACL_rename      equ 0000000100000000b  ; (N) Allow file renaming
ACL_execute     equ 0000000010000000b  ; (E) Allow program execution
ACL_physical    equ 0000000001000000b  ; (P) Allow physical access to device
ACL_attribute   equ 0000000000100000b  ; (A) Allow attribute changing
ACL_indirect    equ 0000000000010000b  ; (I) Allow expansion of indirect files
```

GET USERNAME FROM ACCOUNT FILE

Assembler Interface

INPUT:
AX
5FB2H

BX
Username entry index (0 for first entry, will be updated.)

DS:SI
Pointer to 16-byte buffer to receive username

ES:DI
Pointer to ASCIZ server in the form \\server

OUTPUT:
NC
If no error

CY
If error occurred

AX

Error code if error

BX
Next username entry index (updated by call.)

DS:SI
16-Character username retrieved from server's account file (*not* in ASCIZ form)

C Interface

extern int NOSGetUserAcct(int *npIndex, char *cpUser, char *cpServer);

Pointer to int holding username entry index (updated)

Pointer to buffer to hold user name

Pointer to buffer holding server name

TRANSLATE PATH

TRANSLATE PATH either translates an indirect file into the full network path or it translates the given path into the path relative to the server. This call can be used to determine how an indirect file is going to expand.

For example, a server has a resource named **programs** which contains the following path: **d:\software\programs**. If a user has redirected drive **P:** to **\\server\programs** and the TRANSLATE PATH call is made with DX set to 00000010b, the string **d:\software\programs** is returned.

Note that indirect files are only expanded for a network resource with the indirect (**I**) ACL privilege enabled.

Assembler Interface

INPUT:
AX
5FB3H

DS:SI
Pointer to 128-byte buffer to receive translated path

ES:DI
Pointer to full ASCIZ path including server name. (E.g. \\server\root\prog\file.ext)

DX
Type of translation to be performed. The OR of the following bits:

00000001b Recursively expand indirect files until all are resolved

00000010b Translate to server's physical path. (e.g C:\autoexec.bat)

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

DS:SI
ASCIZ translated path

C Interface

extern int NOSTranslatePath(char *cpPathXlate, char *cpPath, int nType);

Pointer to buffer to receive translated path

Pointer to buffer containing input path

Type of translation to perform

CREATE INDIRECT FILE

CREATE INDIRECT FILE allows you to create an indirect file on a server.

Assembler Interface

INPUT:
AX
5FB4H

DS:SI
Pointer to 128-byte ASCIZ server relative path which will be the indirect file's contents

ES:DI
Pointer to indirect file's full ASCIZ path (e.g. "\\server\root\linkfile")

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSCreateIndir(char *cpPath, char *cpIndir);

Pointer to buffer for path (the DS:SI parameter)

Pointer to buffer with indirect filename (the ES:DI parameter)

Note that a server relative path can be either a path relative to the directory containing the indirect file (for example, **..\dir\file.txt**), or an absolute path beginning with a server resource

name (for example, **\C-DRIVE\dir\file.txt**).

GET INDIRECT FILE CONTENTS

Get indirect file contents will return the contents of an indirect file. This call can be used to determine what the actual contents of an indirect file are.

Assembler Interface

INPUT:
AX
5FB5H

DS:SI
Pointer to 128-byte buffer to receive indirect file contents

ES:DI
Pointer to full network path of indirect file

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

DS:SI
ASCIZ contents of indirect file (a path)

C Interface

extern int NOSGetIndir(char *cpData, char *cpIndir);

Pointer to buffer for file contents

Pointer to buffer with indirect filename

GET SERVER'S TIME

Assembler Interface

INPUT:
AX
5FC0H

DS:SI
Pointer to 8-byte buffer to receive time information

ES:DI
Pointer to ASCIZ server in the form \\server. You must be logged into the server.

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

DS:SI
Filled buffer with time information

C Interface

extern int NOSGetTime(struct time_block *cpTime, char *cpServer);

Pointer to buffer to receive time information

Pointer to buffer with server name
Definition of time buffer:

```
time_block    struc
TB_year       dw  ?   ; Year
TB_day        db  ?   ; Day of month (1-31)
TB_month      db  ?   ; Month (1-12)
TB_minutes    db  ?   ; Minutes (0-59)
TB_hour       db  ?   ; Hour (0-23)
TB_hundredths db  ?   ; Hundredths of seconds (0-99)
TB_seconds    db  ?   ; Seconds (0-59)
time_block    ends
```

SCHEDULE SERVER SHUTDOWN

Assembler Interface

INPUT:
AX
5FC8H

DS:SI
Pointer to 80 character ASCIZ reason string

ES:DI
Pointer to ASCIZ server in the form \\server.

CX
Number of minutes to shutdown (0 is immediate)

DX
Option flags

OUTPUT:
NC

If no error

CY
If error occurred. (You must have the S privilege to execute this operation.)

AX
Error code if error

Note
If a server shutdown is already pending, then subsequent calls will supercede the pending shutdown.

C Interface

extern int NOSShutdown(char *cpReason, char *cpServer, int nMins, int nFlags);

Pointer to buffer containing reason string

Pointer to buffer with server name

Number of minutes to shutdown

Option flags
Shutdown option flags:

```
SHUTDOWN_option_reboot
SHUTDOWN_option_silent     equ 0000000000000010b ; Do not notify users
SHUTDOWN_option_halt       equ 0000000000000100b ; Halt after shutdown
SHUTDOWN_option_powerfail  equ 0000000000001000b ; Shutdown due to power fail
                                ; (Used by UPS.)
SHUTDOWN_option_reserved1  equ 0000000000010000b ; RESERVED
SHUTDOWN_option_reserved2  equ 0000000000100000b ; RESERVED
SHUTDOWN_option_reserved3  equ 0000000001000000b ; RESERVED
SHUTDOWN_option_reserved4  equ 0000000010000000b ; RESERVED
SHUTDOWN_option_user1      equ 0000000100000000b ; User definable
SHUTDOWN_option_user2      equ 0000001000000000b ; User definable
SHUTDOWN_option_user3      equ 0000010000000000b ; User definable
SHUTDOWN_option_user4      equ 0000100000000000b ; User definable
SHUTDOWN_option_user5      equ 0001000000000000b ; User definable
SHUTDOWN_option_user6      equ 0010000000000000b ; User definable
SHUTDOWN_option_user7      equ 0100000000000000b ; User definable
SHUTDOWN_option_reserved5  equ 1000000000000000b ; RESERVED
```

CANCEL SERVER SHUTDOWN

Assembler Interface

INPUT:
AX
5FC9H

ES:DI
Pointer to ASCIZ server in the form \\server.

(You must have the S privilege to use this call.)

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

Note
If no server shutdown is pending, the call will return success.

C Interface

extern int NOSCancelShutdown(char *cpServer);

Pointer to buffer with server name

STUFF SERVER KEYBOARD BUFFER

Assembler Interface

INPUT:
AX
5FCAH

ES:DI
Pointer to ASCIZ server in the form \\server.

(You must have the S privilege to use this call.)

DS:SI
Pointer to 128 byte ASCIZ string to stuff into buffer. (The maximum number of characters that can be stuffed is determined by the server's RUN BUFFER SIZE.)

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSStuffServerBuffer(char *cpText, char * cpServer);

Pointer to string to stuff

Pointer to buffer with server name

GET REDIRECTED PRINTER TIMEOUT

Assembler Interface

INPUT:
AX
5FD0H

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

CX
Redirected printer timeout in ticks (18.2 ticks = 1 second). A value of 0 means timeouts are disabled

C Interface

extern int NOSGetLPTTimeout(int *npTicks);

Pointer to int to set with printer timeout in ticks

SET REDIRECTED PRINTER TIMEOUTS

Assembler Interface

INPUT:
AX
5FD1H

CX
Printer timeout in ticks (18.2 ticks = 1 second). A value of 0 disables the timeout

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSSetLPTTimeout(int nTicks);

Printer timeout in ticks

GET DOS SERVICE VECTOR

The GET and SET DOS SERVICE VECTOR calls allow you to take advantage of the DOS busy checking that the redirector performs. Whenever DOS is safe to be called, the redirector calls the routine pointed to by the SET VECTOR call. This routine can then use any DOS calls without worrying about whether DOS is safe to be called.

To chain into the DOS service vector you must obtain the old vector using GET DOS SERVICE VECTOR and save it. When your service routine is called you need to call the old vector before beginning any processing.

Note

Be careful when using this NOS facility. NOS will call your routine many times a second (potentially thousands.)

Assembler Interface

INPUT:
AX
5FE0H

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

ES:BX
Pointer to current DOS service routine

C Interface

extern int NOSGetDOSVector(FARPROC *fpVector);

Pointer to location that will hold contents of DOS vector

SET DOS SERVICE VECTOR

Assembler Interface

INPUT:
AX
5FE1H

ES:BX
New DOS service vector

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSSetDOSVector(FARPROC Vector);

far address of routine to be called using this vector

GET MESSAGE SERVICE VECTOR

The GET and SET MESSAGE SERVICE VECTOR routines can be used to write a customized message service. To chain into the message service vector you must obtain the old vector using GET MESSAGE SERVICE VECTOR and save it. When your service routine is called you need to call the old vector before beginning any processing.

Note

An example of the use of these calls is included in the sample programs in Chapter 14.

Assembler Interface

INPUT:
AX
5FE2H

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

ES:BX

Pointer to current message service routine

C Interface

extern int NOSGetMsgVector(FARPROC *fpService);

Pointer to far pointer to routine that will hold address of

message service handler

SET MESSAGE SERVICE VECTOR

Assembler Interface

INPUT:
AX
5FE3H

ES:BX
New message service vector location

OUTPUT:
NC
If no error

CY
If error occurred

AX
Error code if error

C Interface

extern int NOSSetMsgVector(FARPROC Service);

Far pointer to routine that will be the new message service handler

Note When a service routine is called ES:BX will point to the currently received message.

tc "Other System Calls" \l2§Other System Calls

tc "Obtaining A List Of Shared Resources" \l3§Obtaining A List Of Shared Resources

The server's shared resources appear as directories or files in the server's conceptual root directory (i.e. **SERVER\.**). Physically, they exist within the server's network control directory. To obtain a list of the server's shared resources you can issue a **findfirst** system call with a path to the server's conceptual network root directory (for example, **server*.***).

For example, to return all shared resources:

```
search_path    db      "\\server\*.*",0
  . . .
  mov  ah, 4eh
; Find first system call
  mov  cx, 11h
; Include read-only & directory
  mov  dx, search_path
  int  21h

  mov  ax, 5fb1h
; Do a Get Shared Directory Information

 ;system call
  mov  si, offset shared_information_buffer
  mov  di, offset <filename result from find first/next call>
  int  21h

  mov  ah, 4fh       ; Find next system call
  int  21h

  . . .              ; \\server has a linkacl whose information can be
                     ;  obtained through the 5FB1H call. Just point
                     ;  es:di at \\server<0>.
```

To return only the printer resources:

```
search_path    db      "\\server\@*.*",0
  . . .
  mov  ah, 4eh       ; Find first system call
```

```
mov  cx, 01h        ; Include read-only files
mov  dx, offset search_path
int  21h

. . .               ; Open found file and read in data. Note that @MAIL
                    ; will be returned as a printer and should be
                    ; masked off.

mov  ah, 4fh        ; Find next system call
int  21h
```

tc "Summary Of Network System Calls" \l2§Summary Of Network System Calls

tc "DOS Compatible Calls" \l3§DOS Compatible Calls

Function Number	C Library Function	Purpose
5E00H	NOSGetMachineName()	Get Machine Name
5E02H	NOSSetPrinterSetup()	Set Printer Setup
5E03H	NOSGetPrinterSetup()	Get Printer Setup
5F02H	NOSGetRedirDevice()	Get Redirected Device Entry
5F03H	NOSRedirDevice()	Redirect Device
5F04H	NOSCancelRedir()	Cancel Device Redirection

tc "LANtastic Network Operating System Calls" \l3§LANtastic Network Operating System Calls

Function Number
C Library Function
Purpose

5F80H
NOSGetLogin()
Get Login Entry

5F81H
NOSLogin()
Login to a Server

5F82H
NOSLogout()
Logout of a Server

5F83H
NOSGetUserName()
Get Username Entry

5F84H
NOSGetServer()
Get Inactive Server Entry

5F85H
NOSChangePassword()
Change Password

5F86H
NOSDisable()
Disable Account

5F87H
NOSGetAccount()
Get Account

5F88H

NOSLogoutAll
Logout from all servers

5F97H
NOSCopyFile()
Copy File

5F98H
NOSSendMsg()
Send Unsolicited Message

5F99H
NOSGetMsg()
Get Last Received Unsolicited Message

5F9AH
NOSGetMsgFlag()
Get Message Processing Flag

5F9BH
NOSSetMsgFlag()
Set Message Processing Flag

5F9CH
NOSPopUpMsg()
Pop Up Last Received Message

5FA0H
NOSGetQueue()
Get Queue Entry

5FA1H
NOSSetQueue()
Set Queue Entry

5FA2H
NOSControlQueue()
Control Queue

5FA3H
NOSGetStatus()
Get Printer Status

5FA4H
NOSGetStreamInfo()
Get Stream Info

5FA5H
NOSSetStreamInfo()
Set Stream Info

5FA7H
NOSCreateAudit()
Create User Audit Entry

5FB0H
NOSGetUserInfo()
Get Active User Information

5FB1H
NOSGetDirInfo()
Get Shared Directory Information

5FB2H
NOSGetUserAcct()
Get Username From Account File

5FB3H
NOSTranslatePath()
Translate Path

5FB4H
NOSCreateIndir()
Create Indirect File

5FB5H
NOSGetIndir()
Get Indirect File Contents

5FC0H
NOSGetTime()
Get Server's Time

5FC8H
NOSShutdown()
Schedule Server Shutdown

5FC9H
NOSCancelShutdown()
Cancel Server Shutdown

5FCAH

NOSStuffServerBuffer()
Stuff Server Keyboard Buffer

5FD0H
NOSGetLPTTimeout()
Get Redirected Printer Timeout

5FD1H
NOSSetLPTTimeout()
Set Redirected Printer Timeouts

5FE0H
NOSGetDOSVector()
Get DOS Service Vector

5FE1H
NOSSetDOSVector()
Set DOS Service Vector

5FE2H
NOSGetMsgVector()
Get Message Service Vector

5FE3H
NOSSetMsgVector()
Set Message Service Vector