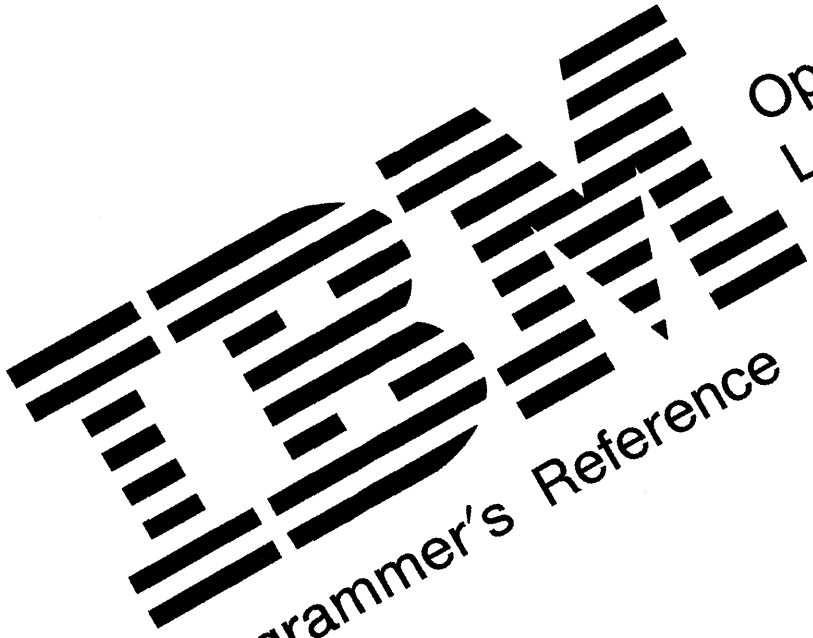

Operating System/2™ LAN Server Version 1.2

Application Programmer's Reference

Programming Family

First Edition (January 1990)

About This Book

This book describes the application programming interface (API) provided by the IBM Operating System/2 Extended Edition Version 1.2 Local Area Network Requester and IBM Operating System/2 Local Area Network Server Version 1.2 program (hereafter referred to as the OS/2 LAN Server), which is a separately available program that works in conjunction with the LAN Requester component of OS/2 Extended Edition Version 1.2 (hereafter referred to as the LAN Requester). The API functions are the external functional interface for application program development.

Who Should Use This Book

This book is intended for use by application programmers and system programmers developing software for use in the IBM OS/2 Extended Edition 1.2 local area network environment.

Before You Use This Book

To use this book effectively, it is recommended that you have a working knowledge of the OS/2 program or some other multitasking operating system and that you are familiar with application programming in IBM C/2™ language (hereafter referred to as C language).

Note: The OS/2 LAN Requester/Server API functions work in essentially the same way for both the IBM OS/2 and IBM DOS operating systems. However, this reference specifically addresses the IBM OS/2 programming environment.

How This Book Is Structured

This book contains the following chapters:

Chapter 1, "Overview of OS/2 LAN Server API," provides an overview of the OS/2 LAN Requester/Server API architecture.

Chapter 2, "Introducing the OS/2 LAN Server API Functions," includes an alphabetical listing of the OS/2 LAN Requester/Server API function categories, with brief descriptions and a list of functions in each category.

Chapter 3, "API Function Descriptions," describes the OS/2 LAN Requester/Server API function categories and the functions in each category. Categories and the functions within them are arranged alphabetically.

Appendix A, "Include Files," lists each of the include files the OS/2 LAN Requester/Server API provides.

Appendix B, "Function Libraries," describes the dynamically linked libraries that OS/2 LAN Requester/Server uses at link and run time.

Appendix C, "Return Codes," describes the API function return codes.

Appendix D, "Creating OS/2 LAN Server Services," provides information on creating and using network service programs.

C/2 is a trademark of International Business Machines Corporation.

Appendix E, "OS/2 LAN Requester/Server API Support under IBM DOS," describes OS/2 LAN Requester/Server API functions supported by IBM DOS along with any differences in their use from that described in Chapter 2.

Appendix F, "IBM C/2 Sample Program," contains a sample program for the OS/2 LAN Server application programming interface.

Appendix G, "PC LAN Program 1.3 Compatibility," provides several function calls for compatibility with applications currently supported by PCLP 1.3.

Appendix H,"LAN API Manifests," lists the manifests associated with ASCIIZ strings that are pointed to by data structure components used in the OS/2 LAN API.

A glossary describes specialized terms used in this book.

An index is included at the back of the book.

Conventions

Throughout this book, the following conventions distinguish elements of text:

Text Element	Use
bold	Command names, switches, and literal portions of syntax that must be written exactly as shown are bold within text paragraphs. (No text styles are applied to elements within monospace entries.)
italics	Variable text representing a type of text to be entered rather than a literal series of characters are in italics within text paragraphs. Italic type is also used to introduce new terms and, occasionally, for emphasis.
`monospace`	Data structures and function syntax templates are monospace, as are any sample program lines within text sections.
CAPITALS	File names and acronyms are in capitals.

Related Publications

The IBM OS/2 LAN Requester/Server documentation set includes other manuals that may be helpful to you. For information on installing, using, or administering OS/2 LAN Requester/Server, consult the following publications:

- *IBM Operating System/2 Local Area Network Server Version 1.2 Getting Started*
- *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide*
- *IBM Operating System/2 Local Area Network Server Version 1.2 User's Guide*
- *IBM Operating System/2 Programming Guide*
- *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference.*

Contents

Chapter 1. Overview of OS/2 LAN Server API

This chapter introduces the OS/2 LAN Requester/Server 1.2 application programming interface (API), which is a set of functions enabling application programs to interact with and to control network operations and resources. In this chapter, you will find information about:

- The organization of and naming conventions for OS/2 LAN Requester/Server API functions
- The API data structures and the way they accept and return information
- The API security scheme
- The protection violations and faults in the dynamic link libraries
- The local and remote function calls
- The requirements for calling API functions
- The naming conventions for network names.

Note: The API functions shown are for use with C language.

Organization of OS/2 LAN Server API Functions

The OS/2 LAN Requester/Server API provides access to the network functions through a well-defined interface for high-level languages. This interface defines the name of a function depending on the task that the function performs.

The OS/2 LAN Requester/Server API naming convention divides each function into three parts:

- The Net, Dos, or Spl keyword identifies the function as an OS/2 LAN Requester/Server API function or an OS/2-compatible function.
- A category identifier indicates the software area in which the function performs. For example, User identifies functions that control user accounts.

  This identifier may be compound, as in ServerDisk, which identifies the function of server disk tasks. Generally, but not always, the identifier correlates to a function category. The notable exception is the Named Pipe category, which contains a broad category of OS/2 functions.
- A verb describes the action the function performs.

An optional fourth part in function names works somewhat as a direct object would in the English language, identifying a particular object to be involved in the action. For instance, NetCharDevQPurgeSelf deletes all pending requests waiting in a serial device queue that were submitted by a particular computer, whereas NetCharDevQPurge deletes all pending requests on a serial device queue.

API Verbs

The OS/2 LAN Requester/Server software defines a set of verbs for each category of functions. The five most common verbs perform the following basic tasks:

Verb	Action
Add	Adds a resource
Del	Deletes a resource
Enum	Lists the names and data structures for a resource
GetInfo	Retrieves parameters for a resource
SetInfo	Modifies parameters for a resource

The less common verbs perform other tasks relating to specific resources, such as starting a service (Install) or deleting pending requests on a queue (Purge).

The five most common OS/2 LAN Requester/Server API verbs use fairly standard syntax and parameters. These operations, which are described in the following sections, comprise the basic set of function tasks for most categories.

Add Functions

An Add function adds a resource to a particular set of items. Add functions generally use some form of the following syntax:

```
unsigned far pascal
NetExampleAdd (servername, level, buf, buflen)
char far *     servername;
short          level;
char far *     buf;
unsigned short buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server (preceded by a double backslash (\\) or a double forward slash (//)) on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies a data structure providing a particular level of detail of information required to complete the operation.
- *buf* points to the data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Del Functions

A Del function removes a resource from a particular set of items. Del functions generally use some form of the following syntax:

```
unsigned far pascal
NetExampleDel (servername)
char far *     servername;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server (preceded by a double backslash (\\) or a double forward slash (//)) on which the function is to execute. A NULL pointer or string specifies a local computer.

Del functions do not require a pointer to a data structure since they do not accept or return the kind of information commonly found in data structures.

An application must have administrative privileges to remotely execute most Del functions.

Enum Functions

The Enum functions list information about system resources. Enum functions generally use some form of the following syntax:

```
unsigned far pascal
NetExampleEnum (servername, level, buf, buflen,
                entriesread, totalentries)
char far *            servername;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  entriesread;
unsigned short far *  totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server (preceded by a double backslash (\\) or a double forward slash (//)) on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail for the returned data.
- *buf* points to the returned data structures.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries available.

If *buf* cannot store all returning data, the Enum function returns the NERR_MORE_DATA error code. There is an *entriesread* number of items returned in the buffer.

If the value of *buflen* is 0, the Enum function returns only a valid *totalentries* parameter.

Some requests for high levels of detail by way of Enum functions require administrative privileges at the remote server.

Enum functions are limited to 64KB per call.

GetInfo Functions

A GetInfo function retrieves specific information about a resource not available to an Enum function. These functions generally use some form of the following syntax:

```
unsigned far pascal
NetExampleGetInfo (servername, level, buf, buflen, totalavail)
char far *           servername,
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server (preceded by a double backslash (\\) or a double forward slash (//)) on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail that the data structure is to return.
- *buf* points to the returned data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information available, given sufficient *buflen.*

If *buf* cannot store all returning fixed-length data, GetInfo returns the NERR_BufTooSmall error code. In this case, all data in *buf* is not valid, but *totalavail* is valid.

If *buf* can store all returning fixed-length data but not all available variable-length data, GetInfo returns the ERROR_MORE_DATA error code. In this case, the fixed-length data in *buf* is valid, with pointers to any incomplete variable-length data set to NULL. If the value of *buflen* is 0, the GetInfo function returns only a valid *totalavail* parameter.

Some requests for high levels of detail by way of GetInfo functions require administrative privileges.

SetInfo Functions

A SetInfo function sets the parameters of a network resource. These functions generally use some form of the following syntax:

```
unsigned far pascal
NetExampleSetInfo (servername, level, buf, buflen, parmnum)
char far *       servername;
short            level;
char far *       buf;
unsigned short   buflen;
short            parmnum;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server (preceded by a double backslash (\\) or a double forward slash (//)) on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail that the data structure is to provide.

- *buf* points to the data structure if *parmnum* is zero. Otherwise, *buf* points to the specific data component that will be changed.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* determines whether an entire data structure or a single data structure component is to be set. If the value is 0, a complete data structure must be provided. Otherwise, *parmnum* should specify the ordinal position of a specific data structure component to be set.

  For information on the ordinal position of a component, see the appropriate include file of the function.

Error Return Codes

In the functional description of each API, there is a list of possible return codes. Following that is a list of low-level functions it calls from which errors might be returned directly. If some errors from an API are ignored or mapped to other APIs, the ignored error codes are listed in brackets [] with a minus sign, such as the following:

```
DosOpen [- ERROR_FILE_NOT_FOUND]
```

This example indicates that DosOpen is called, and all error codes from DosOpen are possibly returned to the caller, except for ERROR_FILE_NOT_FOUND, which is returned by the network API.

The redirector is called by way of special IOCtls and FSCtls, which are represented variously as follows:

```
redir.IOCTLCALLNAME
DosFSCtl(IOCTLCALLNAME)
DosIOCTL(IOCTLCALLNAME)
```

When an API is executed remotely without having the necessary permissions or authorization, it results in the error code ERROR_ACCESS_DENIED. This is a consistent error across all APIs for remote calls. An attempt to execute an API remotely may return ERROR_NETWORK_ACCESS_DENIED, indicating that there was some error in the transportation of the request, rather than an API permission or authorization violation.

API Data Structures

Most API functions use one or more OS/2 LAN Requester/Server-defined data structures to provide or return information defining a resource or reporting its state.

Distinct structures are defined for each category. They use the C language syntax and are WORD-aligned.

Structuring Levels of Detail

The functions in the various API function categories provide different levels of detail, each represented by a different data structure. The more detailed structures usually include all the information in lower-level structures.

When an API function can provide or return more than one level of information, an application must pass a *level* parameter (0, 1, 2, 3 or 10) to indicate the level of detail requested. Level 0 is the least detailed (often a single component), with each subsequent level calling for more detail.

The related *level* parameter is included in the data structure name. For instance, the NetShareGetInfo function uses the *share_info_0*, *share_info_1*, and *share_info_2* data structures for data detailed at the 0, 1, and 2 levels, respectively.

Many functions, particularly the SetInfo type, require a particular data structure specified by a particular level value. If an unacceptable *level* parameter is passed, the function returns the ERROR_INVALID_LEVEL error code.

Sample Data Structures

The three data structures of the NetShareGetInfo function illustrate the levels of detail an API function can provide. The *share_info_0* data structure returns only the netname of a particular resource. NetShareEnum returns a set of these structures—a list containing the names of all resources shared on the server:

```
struct share_info_0 {
       char     shi0_netname[NNLEN+1];
};
```

The *share_info_1* data structure returns the name of the shared resource, parameters indicating the type of resource, and an optional remark provided by way of the NetShareSetInfo function when the share is added on the server (NetShareSetInfo, for obvious reasons, requires a level 1 or 2 data structure).

NetShareEnum, at level 1, returns a set of *share_info_1* structures—one for each resource shared on the server:

```
struct share_info_1 {
  char                    shi1_netname[NNLEN+1];
  char                    shi1_pad1;
  unsigned short          shi1_type;
 char far *               shi1_remark;
};
```

The *share_info_2* data structure adds information on the permissions, path name, number of current uses, and password for a share to the level 1 data structure:

```
struct share_info_2 {
  char                    shi2_netname[NNLEN+1];
  char                    shi2_pad1;
  unsigned short          shi2_type;
  char far *              shi2_remark;
  unsigned short          shi2_permissions;
  unsigned short          shi2_max_uses;
  unsigned short          shi2_current_uses;
  char far *              shi2_path;
  char                    shi2_passwd[SHPWLEN+1];
  char                    shi2_pad2;
};
```

Storing Fixed-Length and Variable-Length Data

When the application is passing a data structure in a buffer to the API, if the data structure has pointers to variable-length data, the buffer length should be the length of the fixed-length portion only. The fixed-length data and the variable-length data do not have to be contiguous in the same memory region, even for remote calls.

When a data structure that defines a pointer to variable-length data (such as ASCIIZ strings) is passed to or returned by a function, an application must provide a buffer

large enough to store both the fixed-length and variable-length data. Otherwise, not all of the data can be passed or returned.

If a buffer is too small to hold all variable-length data associated with a structure, an application should notify the function that no variable-length data is being passed by specifying NULL pointers to the variable-length data.

If an application calls a function that could return more variable-length data than the buffer can store, the function returns as much data as possible, setting all pointers to information that was not returned to NULL. In this case, the function also returns the ERROR_MORE_DATA error code and the number of bytes required to store all available data (the *bytesavail* value). If the buffer is too small for the fixed-length data, the function returns NERR_BusTooSmall or ERROR_MORE_DATA and the amount of data that can fit into the buffer.

API Security Scheme

The OS/2 LAN Server security scheme assigns privilege levels to users. Certain API functions are designated *admin* or *partially admin*. These are available only to users with administrative privileges. These APIs retrieve or set sensitive data or control key network services. OS/2 LAN Requester/Server provides the following types of protection at the API level:

- Administrative privilege
- Security for remote API function calls.

The access control subsystem (ACS) controls the domain-wide access of resources by the users and groups. The users and groups management and resource access control at the API level is called the user accounts subsystem (UAS) database.

Remote Protection

All of the APIs can be executed on a local system, provided that the required software services are running. Ordinary users are treated as administrators locally on the application programming interface.

The API calls to remote servers are subject to privilege checking. Many of the APIs require administrative privilege to run on remote servers.

User Interface and Application Programming Interface

The OS/2 LAN Requester/Server provides user-level security. Share-level security is not supported. The user is required to be logged on in order to run utilities, applications, or user interface programs remotely.

Protection Violations and Faults in the Dynamic Link Libraries

The API functions probe the buffers passed to them and scan string parameters in an attempt to ensure that the data is accessible. These probes may cause faults if the pointers are incorrect (for example, if they are pointing beyond the end of a segment or outside a permitted memory region).

If you get a fault within an OS/2 LAN Requester/Server dynamic link library, attempt to trace the code through the call. By noting the values that are being tested, you can usually recognize the parameter that is causing the problem. Also, check

buffer sizes carefully, since the API functions probe the first and last byte of a buffer even if the data returned or received does not fill the buffer.

If you get a stack overflow, extend the stack size. There is no hard and fast rule for determining the depth of stack that an OS/2 LAN Requester/Server API function requires. Generally, allow 4KB of free stack space for each function call.

Local and Remote Function Calls

All OS/2 LAN Requester/Server API functions can be executed on a local server. Many functions can also be executed on a remote server or a local requester. Functions that can be executed remotely supply the name of an accessible remote server for the *servername* parameter. A NULL *servername* parameter (either a NULL pointer or a NULL string) executes the function locally.

Administrative, Local, and Server APIs

Certain OS/2 LAN Requester/Server API functions can be called only at the administrative, local, or server level. These requirements are noted in parentheses after function titles in the following format:

NetExampleFunction ([*partially*] *admin, local, Server, DOS* [*only*])

These requirements have the following meanings:

API Requirement	Meaning
admin	Can execute remotely only if the calling process has administrative privileges in the domain.
partially admin	Can execute with user privilege on certain level of data structures or user's own information.
local	Can execute only on the local computer.
server	Can execute only on a computer running server software.
DOS	Can execute both under OS/2 and DOS requesters.

When administrative privilege for an operation is inadequate, the function returns the error message ERROR_ACCESS_DENIED.

DOS LAN Requester Considerations

When porting OS/2 LAN Requester/Server applications to run under DOS, note that DOS, unlike the OS/2 program, does not support pointer checking, semaphores, or shared memory segments. Also, note that all file names, directory names, or parts of a path name, including UNC server and network names, must follow DOS naming conventions.

Network Naming Conventions

The OS/2 LAN Requester/Server API defines name formats (ASCIIZ strings) to distinguish various parts of the network software. Thus, an API function can easily distinguish the type of resource or device parameter that is being passed.

The server name must be preceded by a double back or forward slash (\\ or //).

The format and maximum length of each type of name are defined in the NETCONS.H include file.

The OS/2 LAN Server Version 1.2 API supports OS/2 Extended Edition Version 1.2 file names. For OS/2 LAN Server version 1.2, the OS/2 program limits fully specified paths to 260 characters, including the following:

- The drive letter
- The colon (:)
- All of the characters in the path name, including all backslashes (\) and slashes (/)
- The file or directory name on the end of the path
- The null character on the end of the path.

  For OS/2 LAN Server Version 1.2, the operating system limits component names to 255 characters. A *component name* is a file name or directory name (or a psuedo directory name). It is the part of a path between the two backslashes or between a slash and the null character on the end of the path. The 255 characters include all of the characters in the component name, but do not include the backslashes or the ending null character.

All characters can be used in network names except ASCII characters less than hexadecimal 20 and the following:

```
" / [ ] : | < > + = ; ,
```

Spaces are not allowed in domain names.

Periods can be used; however, they cannot be the first character of a network name or immediately follow another period in a name. For example,

```
work.sta.1
```

is valid because a period does not start the name and the second period does not immediately follow the first use of a period. But

```
.work.sta.1
```

and

```
work..sta.1
```

are not valid because of a period at the beginning of a name and the two periods used together.

Chapter 2. Introducing the OS/2 LAN Server API Functions

This chapter contains a categorical list of the OS/2 LAN Requester/Server API functions, giving a brief description of the action performed by each function.

For detailed descriptions of the OS/2 LAN Requester/Server API functions, see Chapter 3, "API Function Descriptions."

Function Categories

Twenty-four categories of API functions perform various OS/2 LAN Requester/Server network tasks. For example, the Serial Device category contains all functions that are used to control shared serial devices. In the descriptions that follow, both the categories and the functions within each category are listed alphabetically. The function name is followed by an italicized label in parentheses, which describes when the function can be successfully called. For example, the NetShareAdd function can be executed only by an application with administrative privileges on a server.

Note: The function names are shown as they should be used in a C language program; that is, the function names must be entered in uppercase and lowercase letters.

Access Permission

The functions in the Access Permission category examine or modify user or group access permission records for server resources.

Function	Description
NetAccessAdd (*admin, DOS*)	Creates an access permission record assigning user and group permissions for a new resource.
NetAccessCheck (*local*)	Verifies a user's or group's permission to access a particular resource.
NetAccessDel (*admin, DOS*)	Deletes all access permission records for a particular shared resource.
NetAccessEnum (*admin, DOS*)	Enumerates all access permission records for a particular server resource.
NetAccessGetInfo (*admin, DOS*)	Retrieves information about an access permission record for a resource.
NetAccessGetUserPerms (*partially admin, DOS*)	Supplies a specified user or group permission for a resource.
NetAccessSetInfo (*admin, DOS*)	Modifies an access permission record for a resource.

Alert

The functions in the Alert category provide a system for notifying network service programs and applications of network events.

Function	Description
NetAlertRaise (*local*)	Notifies all clients registered in the alert table that a particular event occurred.
NetAlertStart (*local*)	Registers a client to be notified of a particular type of network event.
NetAlertStop (*local*)	Removes a client registration from an alert table.

Auditing

The functions in the Auditing category control the audit log file, which contains an audit trail of operations that occur on a server.

Function	Description
NetAuditClear (*admin, DOS*)	Clears (and optionally saves) the audit log file of a server.
NetAuditRead (*admin, DOS*)	Opens and returns an OS/2 file handle to the audit log file of a server.
NetAuditWrite (*local, server*)	Writes an audit trail entry to the local audit log file.

Configuration

The functions in the Configuration category retrieve network configuration information from the IBMLAN.INI file.

Function	Description
NetConfigGet2 (*admin, DOS*)	Retrieves a specified parameter value for a given network component in the IBMLAN.INI file from a remote computer.
NetConfigGetAll2 (*admin, DOS*)	Retrieves all parameter information for a given network component in the IBMLAN.INI file from a remote computer.

Connection

The NetConnectionEnum function gives a listing of all connections made to a server by a requester client or all connections made to the shared resource of a server.

Function	Description
NetConnectionEnum (*admin, server, DOS*)	Lists either all connections between requesters and resources on a server or all connections established within a session.

Domain

The functions in the domain category provide domain-wide information.

Function	Description
NetGetDCName (*DOS*)	Obtains the name of the domain controller.
NetLogonEnum (*partially admin, DOS*)	Supplies information about logged on users.

Error Logging

The functions in the Error Logging category control the error log file.

Function	Description
NetErrorLogClear (*admin*)	Clears (and optionally saves) an error log file.
NetErrorLogRead (*admin*)	Opens and returns an OS/2 file handle to the error log file of a computer.
NetErrorLogWrite (*local*)	Writes an entry to the error log file of a computer.

File

The functions in the File category provide a system for monitoring the file, device, and pipe resources that are opened on a server, and for closing one of these resources if necessary.

Function	Description
NetFileClose2 (*admin, server, DOS*)	Forces a resource closed when a system error prevents a normal DosClose function closing.
NetFileEnum2 (*admin, server, DOS*)	Allows the user to issue iterated calls to get information about some or all open files on a server.
NetFileGetInfo2 (*admin, server, DOS*)	Retrieves information about a particular opening of a server resource.

Group

The functions in the Group category control user groups in the user accounts subsystem (UAS) database.

Function	Description
NetGroupAdd (*admin, DOS*)	Creates a new group account.
NetGroupAddUser (*admin, DOS*)	Adds a user to a group.
NetGroupDel (*admin, DOS*)	Removes a group account from the UAS database.
NetGroupDelUser (*admin, DOS*)	Removes a user from a particular group.

Function	Description
NetGroupEnum (*partially admin, DOS*)	Lists all group accounts.
NetGroupGetInfo (*partially admin, DOS*)	Obtains group-related information.
NetGroupGetUsers (*partially admin, DOS*)	Lists the members of a particular group.
NetGroupSetInfo (*admin, DOS*)	Sets group-related information.
NetGroupSetUsers (*admin, DOS*)	Sets information about users who belong to a group.

Handle

The functions in the Handle category obtain and set information on a per-handle basis.

Function	Description
NetHandleGetInfo (*local, server*)	Obtains handle-specific information.
NetHandleSetInfo (*local, server*)	Sets information in the data structure of a handle.

Mailslot

The functions in the Mailslot category provide one-way interprocess communication (IPC).

Function	Description
DosDeleteMailslot (*local, DOS*)	Deletes a mailslot, discarding all messages, whether or not they have been read.
DosMailslotInfo (*local, DOS*)	Returns information about a particular mailslot.
DosMakeMailslot (*local, DOS*)	Creates a mailslot and returns its handle.
DosPeekMailslot (*local, DOS*)	Reads the next message in a mailslot without removing any data.
DosReadMailslot (*local, DOS*)	Reads, then removes the most current message from a mailslot (based on priority).
DosWriteMailslot (*local, DOS*)	Writes a message to a particular mailslot.

Message

The functions in the Message category are used to send, log, and forward messages.

Function	Description
NetMessageBufferSend (*admin, DOS*)	Sends a buffer of information to a registered user on a particular computer.
NetMessageFileSend (*admin, DOS*)	Sends a file to a registered user on a particular computer.
NetMessageLogFileGet (*admin, DOS*)	Retrieves the name of the message log file and the current logging status (on or off).
NetMessageLogFileSet (*admin, DOS*)	Specifies a file to log messages received by registered users and enables or disables logging.
NetMessageNameAdd (*admin, DOS*)	Registers a user in the message-name table.
NetMessageNameDel (*admin, DOS*)	Deletes a user name from a message-name table.
NetMessageNameEnum (*admin, DOS*)	Lists the user name entries in a message-name table.
NetMessageNameFwd (*admin*)	Modifies the message-name table to forward a user's messages to another user.
NetMessageNameGetInfo (*admin, DOS*)	Retrieves information about a user's message account.
NetMessageNameUnFwd (*admin*)	Stops forwarding a user's messages to another user.

Named Pipe

The functions in the Named Pipe category control interprocess communication (IPC) for named pipes. These functions are provided by the base operating system and supported by the OS/2 LAN Server across the network.

Function	Description
DosBufReset (*local, DOS*)	Clears the data buffer of a named pipe.
DosCallNmPipe (*local, DOS*)	Opens a named pipe, performs a write to the pipe followed by a read, and then closes the pipe.
DosClose (*local, DOS*)	Closes a named pipe.
DosConnectNmPipe (*local*)	Waits for a client process to open an instance of a named pipe.
DosDisconnectNmPipe (*local*)	Forces a named pipe to close, denying a client process any further access to it.
DosDupHandle (*local, DOS*)	Duplicates the handle to a named pipe.

Function	Description
DosMakeNmPipe (*local*)	Creates a new named pipe or a new instance of an existing named pipe and returns its handle.
DosOpen (*local, DOS*)	Opens the client process end of a named pipe and returns a handle.
DosPeekNmPipe (*local, DOS*)	Reads the data in a named pipe without removing it.
DosQFHandState (*local, DOS*)	Retrieves information about whether the handle of a named pipe is inheritable and whether write-behind to remote pipes is allowed.
DosQHandType (*local, DOS*)	Returns the type of a particular handle.
DosQNmPHandState (*local, DOS*)	Returns information about the current state of a named pipe.
DosQNmPipeInfo (*local, DOS*)	Retrieves information about the sizes of the incoming and outgoing buffers of a named pipe and the number of instances that are available.
DosQNmPipeSemState (*local*)	Returns information about the status of a semaphore associated with a named pipe on a local computer.
DosRead (*local, DOS*)	Reads data from a named pipe.
DosReadAsync (*local*)	Reads data from a named pipe asynchronously, removing the data.
DosSetFHandState (*local, DOS*)	Modifies the open mode state of a named pipe.
DosSetNmPHandState (*local, DOS*)	Modifies the read mode and blocking mode state of a named pipe.
DosSetNmPipeSem (*local*)	Associates a semaphore with the client or server process of a local named pipe.
DosTransactNmPipe (*local, DOS*)	Writes a message to and then reads a message from a named pipe.
DosWaitNmPipe (*local, DOS*)	Enables a client process to wait for an available instance of a named pipe.
DosWrite (*local, DOS*)	Writes data to a file or named pipe.
DosWriteAsync (*local*)	Writes data to a named pipe asynchronously.

Remote Utility

The functions in the Remote Utility category enable applications to copy and move remote files, remotely execute a program, and access the time-of-day information on a remote server.

Function	Description
NetRemoteCopy (*local, DOS*)	Copies one or more files from one location to another.
NetRemoteExec (*local, server*)	Executes a program located on a remote server.
NetRemoteMove (*local, DOS*)	Moves one or more files from one location to another.
NetRemoteTOD (*DOS*)	Returns time of day on a server.

Requester

The functions in the Requester category control the operation of requesters.

Function	Description
NetWkstaGetInfo (*partially admin, DOS*)	Returns information about the configuration components of a requester.
NetWkstaSetInfo (*admin, DOS*)	Configures a requester.
NetWkstaSetUID2 (*admin, DOS*)	Registers a user name and password with the redirector to validate the user account.

Serial Device

The functions in the Serial Device category control shared serial devices and their associated queues.

Function	Description
NetCharDevControl (*admin, server, DOS*)	Forces a serial device to close.
NetCharDevEnum (*admin, server, DOS*)	Lists all serial devices in a shared serial device queue on a server.
NetCharDevGetInfo (*server, DOS*)	Retrieves information about a particular serial device in a shared serial device queue on a server.
NetCharDevQEnum (*server, DOS*)	Lists all serial device queues on a server.
NetCharDevQGetInfo (*server, DOS*)	Retrieves information about a particular serial device queue on a server.
NetCharDevQPurge (*admin, server, DOS*)	Deletes all unprocessed requests on a serial device queue.
NetCharDevQPurgeSelf (*server, DOS*)	Deletes all pending requests waiting in a serial device queue submitted by a particular computer.

Function	Description
NetCharDevQSetInfo (*admin, server, DOS*)	Modifies the state of a serial device queue on a server.

Server

The functions in the Server category enable remote administrative tasks to be performed on a local or remote server.

Function	Description
NetServerAdminCommand (*admin, server, DOS*)	Executes a command on a server.
NetServerDiskEnum (*admin, DOS*)	Retrieves a list of local disk drives on a computer.
NetServerEnum2 (*DOS*)	Enumerates the set of all machine IDs visible on the network.
NetServerGetInfo (*partially admin, server, DOS*)	Retrieves information at one of four levels of detail about a particular server.
NetServerSetInfo (*admin, server, DOS*)	Sets the operating parameters for a server.

Service

The functions in the Service category start and control network service programs.

Function	Description
NetServiceControl (*partially admin, DOS*)	Controls the operations of network services.
NetServiceEnum (*DOS*)	Retrieves information about all network services started on a server or a requester.
NetServiceGetInfo	Retrieves information about a particular started network service.
NetServiceInstall (*admin, DOS*)	Starts a network service on a server.
NetServiceStatus	Sets status and code information for a network service.

Session

The functions in the Session category control network sessions established between requesters and servers.

Function	Description
NetSessionDel (*admin, server, DOS*)	Ends a session between a requester and a server.
NetSessionEnum (*partially admin, server, DOS*)	Provides information on all current sessions to a server.

Function	Description
NetSessionGetInfo (*partially admin, server, DOS*)	Retrieves information about a session established between a particular requester and server.

Share

The functions in the Share category control shared resources.

Function	Description
NetShareAdd (*admin, server, DOS*)	Creates a shareable resource.
NetShareCheck (*server, DOS*)	Queries whether a server is sharing a device.
NetShareDel (*admin, server, DOS*)	Deletes a netname from shared resources.
NetShareEnum (*partially admin, server, DOS*)	Retrieves share information about each shared resource.
NetShareGetInfo (*partially admin, server, DOS*)	Retrieves information about a particular shared resource.
NetShareSetInfo (*admin, server, DOS*)	Sets a new share parameter or parameters for a shared resource.

Spooler

The functions in the Spooler category provide applications access to spooler queue manager operations. These functions are provided by the base operating system and supported by the OS/2 LAN Server across the network.

Function	Description
SplQmAbort	Stops the generation of the spool files and automatically closes the spooler queue manager.
SplQmClose	Closes the spooler queue manager.
SplQmEndDoc	Ends a print job and returns a unique job number.
SplQmOpen	Opens the spooler queue manager for generating a print job.
SplQmStartDoc	Signifies the start of a print job.
SplQmWrite	Writes a buffer to the spool file for the print job.

Statistics

The functions in the Statistics category retrieve and clear the operating statistics for requesters and servers.

Function	Description
NetStatisticsGet2 (*admin, DOS*)	Obtains and optionally clears the operating statistics for a server.

Use

The functions in the Use category examine or control connections (uses) between requesters and servers.

Function	Description
NetUseAdd (*admin, DOS*)	Establishes a connection between a local or NULL device name and a shared resource by redirecting the local or NULL universal naming convention (UNC) device name to the shared resource.
NetUseDel (*admin, DOS*)	Ends a connection between a local or UNC device name and a shared resource.
NetUseEnum (*admin, DOS*)	Lists all current connections between the local requester and resources on a remote server.
NetUseGetInfo (*admin, DOS*)	Retrieves information about a connection between a local device and a shared resource.

User

The functions in the User category control a user's account in the UAS database.

Function	Description
NetUserAdd (*admin, DOS*)	Adds a user to the set of those permitted to use the resources of a server.
NetUserDel (*admin, DOS*)	Removes a user's account from the UAS database, ending the user's access to the resources of the server.
NetUserEnum (*partially admin, DOS*)	Returns information about all user accounts.
NetUserGetGroups (*partially admin, DOS*)	Lists all groups on a server to which a particular user belongs.
NetUserGetInfo (*partially admin, DOS*)	Retrieves information about a particular user account.

Function	Description
NetUserModalsGet (*partially admin, Dos*)	Obtains global modals-related information for all users and groups in the UAS database.
NetUserModalsSet (*admin, DOS*)	Sets global modals-related information for all users and groups in the UAS database.
NetUserPasswordSet (*DOS*)	Changes the password in a user's account.
NetUserSetGroups (*admin, DOS*)	Sets the groups of which a user is member.
NetUserSetInfo (*partially admin, DOS*)	Modifies permission information about a particular user name.
NetUserValidate2 (*local*)	Validates a user ID with its password and verifies that the user can log on based on logon restrictions defined for the account.

Chapter 3. API Function Descriptions

This chapter provides detailed information about the syntax of each API function, the tasks that the function performs, and data structures and header files that it uses. This chapter includes twenty-four reference sections, each representing one of the twenty-four function categories outlined in Chapter 2, "Introducing the OS/2 LAN Server API Functions."

Format of API Reference Pages

Each function category begins with an overview explaining how the functions interrelate and how they work with the network software. Any data structures common to several or all of the functions are then described. Each category contains a separate reference section for each function. In the reference section, syntax is described and parameters are defined. Error codes returned by the function are listed and briefly described. In most cases, a discussion expanding on function requirements or behaviors not previously covered is also included.

Both the function categories and the function reference sections within them are ordered alphabetically.

The following subsections describe the type of information that can be found on the individual API reference pages:

Title Briefly describes why, how, or when your application should use the function. The function title is followed by a brief description of the use of the function. A complete list of function restrictions is described in Chapter 2, "Introducing the OS/2 LAN Server API Functions."

Syntax Describes the header files that must be included before your application calls the function. In addition, it provides the definition of the function and a detailed description of the parameters of each function.

Return Codes Provides a list of the return codes the function is most likely to return. This list is not exhaustive. Low-level operating system conditions may return other codes. For a complete listing of OS/2 LAN Requester/Server error codes, see Appendix C, "Return Codes."

Remarks Describes important details about the performance of the function or any peculiarities or special behaviors of the function that your application should take into consideration in order to successfully call the function and efficiently use its results.

Related Information References other sections or chapters in this or other manuals in the OS/2 LAN Requester/Server document set that may help you better understand or use the function.

Access Permission Category

NetAccessAdd (*admin, DOS*)—See "NetAccessAdd" on page 3-6.

NetAccessCheck (*local*)—See "NetAccessCheck" on page 3-9.

NetAccessDel (*admin, DOS*)—See "NetAccessDel" on page 3-12.

NetAccessEnum (*partially admin, DOS*)—See "NetAccessEnum" on page 3-15.

NetAccessGetInfo (*partially admin, DOS*)—See "NetAccessGetInfo" on page 3-18.

NetAccessGetUserPerms (*partially admin, DOS*)—See "NetAccessGetUserPerms" on page 3-22.

NetAccessSetInfo (*admin, DOS*)—See "NetAccessSetInfo" on page 3-25.

The functions in the Access Permission category examine or modify user or group access permission records for server resources. They are used with the ACCESS.H and NETCONS.H include files.

Description

In order for a user to access a shared resource, an *access permission record* must be defined for that user. An access permission record defines how a user or group can access a shared resource. It contains a set of permissions for each user or group.

Access permission records are created using the NetAccessAdd function. To delete all access permission records associated with a particular shared resource, call NetAccessDel.

An access permission record contains:

- The name of the resource
- A list of users or groups permitted to use the resource
- A list of access permissions granted to a particular user or group.

The NetAccessGetInfo function can be called to return information on a particular access permission record. To obtain information on all access permission records for which the calling process has special permissions (ACCESS_PERM), call NetAccessEnum.

The access permission record must be defined by a user or application that already has administrative permissions , or has special permission (ACCESS_PERM) for the resource being shared. Note that user permissions have precedence over group permissions. If a user is not defined in the access list for the shared resource, the user's access permissions are the union of all groups to which the user belongs. For more information on access control checking, see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide*. The NetAccessCheck function can be called to verify whether a user has permission to access a particular resource. If the user or group does not have access permission and access permission is needed, the access permission record can be changed from its original content with NetAccessSetInfo.

DOS Considerations

Under DOS, these functions can be executed only on a remote server. Administrative privilege must have been granted to execute the functions. Attempting to execute the functions on a local requester returns NERR_RemoteOnly.

Data Structures

The *level* parameter controls the level of information provided to or returned from the NetAccessAdd, NetAccessEnum, NetAccessGetInfo, and NetAccessSetInfo functions. These functions use either a level 0 or a level 1 data structure.

Access Permission Information (Level 0)

```
struct access_info_0 {
    char far * acc0_resource_name;
};
```

where:

- *acc0_resource_name* points to an ASCIIZ string containing the name of a resource type. *acc0_resource_name* uses the following formats:

Resource Type	Name Format
Directory	*drive:pathname*
File	*drive:pathname*
Pipe	**\pipe***pipename*
Spooler queue	**\print***queuename*
Serial device queue	**\comm***chardevqueue*

Access Permission Information (Level 1)

```
struct access_info_1 {
    char far * acc1_resource_name;
    short      acc1_attr;
    short      acc1_count;
};
```

where:

- *acc1_resource_name* points to an ASCIIZ string specifying the name of a particular resource (see preceding discussion on *acc0_resource_name*).
- *acc1_attr* specifies the attributes of *acc1_resource_name*. The bits of *acc1_attr* are defined as follows:

Bit	Meaning
0	Audit all. When this bit is set, all access attempts will be audited. No other bits in the field can be set. It is an error to set any other bits when bit 0 is set. When bit 0 is cleared, the remaining bits are defined as described as follows in this table.
1-3	Reserved with a value of 0.
4	If 1, audit successful file opens.

Bit	Meaning
5	If 1, audit successful file writes and successful directory creates.
6	If 1, audit successful file deletes or truncates and successful directory deletes.
7	If 1, audit successful file and directory ACL changes.
8	If 1, audit failed file opens.
9	If 1, audit failed file writes and failed directory creates.
10	If 1, audit failed file deletes or truncates and failed directory deletes.
11	If 1, audit failed file and directory ACL changes.
12-15	Reserved with a value of 0.

Notes:

1. Other resources that can be accessed across the network, including spooler queues, serial device queues, and pipes, are audited using the FOR FILES bits.
2. A value of 0 for the accl_attr word means that there is no auditing of resource accesses. A value of 1 means audit everything. Other values indicate the auditing of specific accesses.
3. When write auditing is enabled, the "write audit" record will be generated when the file is successfully opened for write. written. Only one "write audit" record is produced per open instance of the file. If both write and open auditing are enabled, two audit records may be produced.
4. File size changes (including truncation) are audited under the control of auditing bits 5 and 9. Thus, access that is controlled with the ACCESS_WRITE permission bits is audited by way of auditing bits 5 and 9.
5. Bit 3 is used in conjunction with bit 4 to allow the auditor to determine the duration of access. However, since this information is not required, the generation of the close audit is optional.

- *accl_count* specifies the number of *access_list* data structures following the *access_info_1* data structure.

In addition, the *access_info_1* data structure can be followed by zero or more (up to a maximum of 64) *access_list* data structures. These structures are used to define resource permissions for individual users or groups.

Resource Permissions

```
struct access_list {
    char  acl_ugname[UNLEN+1];
    char  acl_ugname_pad_1;
    short acl_access;
};
```

where:

- *acl_ugname* is an ASCIIZ string specifying a particular user name or group name.
- *acl_ugname_pad_1* WORD-aligns the data structure components.
- *acl_access* specifies permission of a user name or a group name. *acl_access* is defined in ACCESS.H as follows:

Manifest	Bit Mask	Meaning
ACCESS_READ	0x01	Permission to read data from a resource, and by default execute the resource.
ACCESS_WRITE	0x02	Permission to write data to the resource.
ACCESS_CREATE	0x04	Permission to create an instance of the resource (such as a file); data can be written to the resource when creating it.
ACCESS_EXEC	0x08	Permission to execute the resource.
ACCESS_DELETE	0x10	Permission to delete the resource.
ACCESS_ATRIB	0x20	Permission to modify the attributes of a resource (such as the date and time a file was last modified).
ACCESS_PERM	0x40	Permission to modify the permissions (read, write, create, execute, and delete) assigned to a resource for a user or application.
ACCESS_ALL	0x7F	Permission to read, write, create, execute, or delete a resource, or to modify attributes or permissions.
ACCESS_GROUP	0x8000	Permission for a particular group; if returned, indicates that the entry is for a group.

Related Information

For information on include files, see Appendix A, "Include Files."

NetAccessAdd

The NetAccessAdd (admin, DOS) function defines a user name or group name access permission record for a new resource.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetAccessAdd(servername, level, buf, buflen)
char far *     servername;
short          level;
char far *     buf;
unsigned short buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail (1) provided in the *access_info_1* data structure.
- *buf* points to the *access_info_1* data structure. The structure can be followed by zero or more *access_list* data structures.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_BAD_NETPATH	53	The network path cannot be found.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ResourceNotFound	2222	The netname cannot be found.
NERR_ResourceExists	2225	The resource permission list already exists.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFNoRoom	2228	There are too many names in the access control file.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.

Manifest	Value	Meaning
NERR_ACFTooManyLists	2230	Too many lists were specified.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosFreeSeg
- DosFsCtl
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosFSRamSemClear
- DosGetShrSet[-ERROR_FILE_NOT_FOUND]
- DosOpen
- DosQFileMode
- DosSemClear.

Remarks

To define the access permissions for a new resource, the contents of *buf* must include an *access_info_1* data structure specifying the name of a resource, attributes, and the number of *access_list* data structures that are appended. Each *access_list* data structure specifies a user name or group name and associated permissions to be added to the access permission record of a resource.

Related Information

For information on:

- Add functions—See Chapter 1, "Overview of OS/2 LAN Server API."
- Deleting an access permission record—See "NetAccessDel" on page 3-12.
- Listing server permissions and resources—See "NetAccessEnum" on page 3-15.

NetAccessCheck

The NetAccessCheck (local) function verifies that a user name has the supplied permissions for a particular resource.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetAccessCheck(reserved, uname, resource, operation, result)
char far *           reserved;
char far *           uname;
char far *           resource;
unsigned short       operation;
unsigned short far *  result;
```

where:

- *reserved* is a NULL pointer.
- *uname* points to an ASCIIZ string containing a user name.
- *resource* points to an ASCIIZ string containing the local path name for the resource type, as follows:

Type	Format
Directory	*drive:pathname*
File	*drive:pathname*
Pipe	**\pipe***pipename*
Spooler queue	**\print***queuename*
Serial device queue	**\comm***chardevqueue*

- *operation* specifies the type of access operation requested. Any combination of the following can be requested, as defined in ACCESS.H as follows:

Manifest	Bit Mask	Meaning
ACCESS_READ	0x01	Permission to read data from a resource, and by default execute the resource.
ACCESS_WRITE	0x02	Permission to write data to the resource.
ACCESS_CREATE	0x04	Permission to create an instance of the resource (such as a file); data can be written to the resource when creating it.
ACCESS_EXEC	0x08	Permission to execute the resource.
ACCESS_DELETE	0x10	Permission to delete the resource.
ACCESS_ATRIB	0x20	Permission to modify the attributes of a resource (such as the date and time a file was last modified).

Manifest	Bit Mask	Meaning
ACCESS_PERM	0x40	Permission to modify the permissions (read, write, create, execute, and delete) assigned to a resource for a user or application.
ACCESS_ALL	0x7F	Permission to read, write, create, execute, or delete a resource, or to modify attributes or permissions.

- *result* points to an unsigned short integer specifying whether or not the operation is permitted. *result* is only valid when the NetAccessCheck function returns the NERR_Success return code. If *result* is 0, then the operation is permitted.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_SEEK	25	The seek is invalid.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_InternalError	2140	An internal error has occurred.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.

Other error return codes may be returned from the following functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsCtl
- DosOpen
- DosRead.

Remarks

If an access permission record cannot be found for the specified user name and the specified resource, the NetAccessCheck function tries to find the proper access permission record for the GUEST account, a special account set up for temporary users of the resource. GUEST accounts are defined in the IBMLAN.INI file.

Related Information

For information on:

- Defining user or group access permissions—See "NetAccessAdd" on page 3-6.
- Guest accounts—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- Listing all permissions and resources—See See "NetAccessEnum" on page 3-15.

NetAccessDel

The NetAccessDel (admin, DOS) function deletes all access permission records for a particular shared resource.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetAccessDel(servername, resource)
char far *   servername;
char far *   resource;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *resource* points to an ASCIIZ string containing the local path name for the resource type, as follows:

Type	Format
Directory	*drive:pathname*
File	*drive:pathname*
Pipe	**\pipe***pipename*
Spooler queue	**\print***queuename*
Serial device queue	**\comm***chardevqueue*

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_ResourceNotFound	2222	The netname cannot be found.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFSRamSemClear
- DosFreeSeg
- DosFsCtl
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)

- DosFsctl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosOpen
- DosSemClear.

Related Information

For information on:

- Defining user name or group name access permissions—See "NetAccessAdd" on page 3-6.
- Del functions—See Chapter 1, "Overview of OS/2 LAN Server API."
- Listing all permissions and resources—See See "NetAccessEnum" on page 3-15.

NetAccessEnum

The NetAccessEnum (partially admin, DOS) function enumerates all access permission records.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetAccessEnum(servername, basepath, recursive, level,
                      buf, buflen, entriesread, totalentries)
char far *            servername;
char far *            basepath;
short                 level;
short                 recursive;
char far *            buf;
unsigned short        buflen;
unsigned short far *  entriesread;
unsigned short far *  totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *basepath* points to an ASCIIZ string containing a base path name for the shared resources. A NULL pointer or string means no *basepath* is to be used.
- *recursive* enables or disables recursive searching. If *recursive* is 0, NetAccessEnum returns entries only for the resource named as *basepath*. If *recursive* is non-zero, NetAccessEnum returns entries for all access control records whose resource matches *basepath*.
- *level* specifies the level of detail (0 or 1) requested for the returned *access_info* data structure.
- *buf* points to the returned *access_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries returned.
- *totalentries* points to an unsigned short integer indicating the number of entries available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.

Manifest	Value	Meaning
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ResourceNotFound	2222	The netname cannot be found.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFNoRoom	2228	There are too many names in the access control file.

Manifest	Value	Meaning
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following functions:

- DosAllocSeg
- DosFsRamSemClear
- DosFreeSeg
- DosFsctl(NETTRANSACTION)
- DosFsctl(NULLTRANSACT)
- DosFsctl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosOpen
- DosSemClear.

Remarks

The NetAccessEnum function can return entries for an application having only ACCESS_PERM permissions. If the user does not have administrative privileges, NetAccessEnum will not return ERROR_ACCESS_DENIED, but it will return NERR_Success with zero entries. The *basepath* parameter limits the entries returned by NetAccessEnum. If *basepath* is a non-NULL string, *basepath* serves as a prefix for the path name. For example, if *basepath* is C:\PROG, NetAccessEnum returns access permission records for resources that begin with C:\PROG.

The *totalentries* parameter indicates the number of entries available for the given *basepath* and *recursive* parameters, not the total number of entries in the access file.

Therefore, NetAccessEnum returns information only for resources with non-default settings below the root directory specified in the request. Note that this is semantically consistent with a standard OS/2 LAN Requester/Server, which returns only explicit permissions. In addition, it is highly recommended that the recursive switch always be set to FALSE.

Related Information

For information on:

- Adding an access permission record—See "NetAccessAdd" on page 3-6.
- Enum functions—See Chapter 1, "Overview of OS/2 LAN Server API."
- Retrieving information about a permissions of a resource—See "NetAccessGetInfo" on page 3-18.
- Verifying an access permission record of a resource—See "NetAccessCheck" on page 3-9.

NetAccessGetInfo

The NetAccessGetInfo (partially admin, DOS) function retrieves information about the access permission record of a resource.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetAccessGetInfo(servername, resource, level, buf,
                      buflen, totalavail)
char far *            servername;
char far *            resource;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *resource* points to an ASCIIZ string containing the local path name for the resource type, as follows:

Type	Format
Directory	*drive:pathname*
File	*drive:pathname*
Pipe	**\pipe***pipename*
Spooler queue	**\print***queuename*
Serial device queue	**\comm***chardevqueue*

- *level* specifies the level of detail (0 or 1) requested for the returned *access_info* data structure.
- *buf* points to the returned *access_info* data structure. On a successful return, *buf* can contain an *access_info_0* data structure or an *access_info_1* data structure followed by zero or more *access_list* data structures. The number of *access_list* data structures returned can be found in the *accl_count* component of the *access_info_1* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_ResourceNotFound	2222	The netname cannot be found.

Manifest	Value	Meaning
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosFsRamSemClear
- DosFreeSeg
- DosFsctl
- DosFsctl(NETTRANSACTION)
- DosFsctl(NULLTRANSACT)
- DosFsctl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosOpen
- DosSemClear.

Remarks

If the calling process does not have administrative privileges, NetAccessGetInfo can be successfully called only by a process that has special permissions (ACCESS_PERM) defined in the access permission record of the resource.

The specified *resource* must be a complete path name.

In specifying the *queuename* for a resource, use the name originally assigned to the resource with the NetShareAdd function.

If *level* is 1, NetAccessGetInfo returns an *access_info_1* data structure followed by an *access_list* data structure for each entry in the list of the resource. The number of entries can be determined by examining the *acc1_count* component in the *access_info_1* data structure.

If *buf* cannot hold all of the fixed-length and variable-length data (all *access_list* data structures), NetAccessGetInfo returns the NERR_BufTooSmall error code, and not the ERROR_MORE_DATA error code as most GetInfo functions do when there is more data available.

The proper way to determine the size of *buf* is to first call NetAccessGetInfo with *level* as 1 and *buflen* as 0. In this case, NetAccessGetInfo will return the number of bytes required in *totalavail*. After obtaining this value, call NetAccessGetInfo with *level* as 1, and specify the new *buflen*.

Related Information

For information on:

- GetInfo functions—See Chapter 1, "Overview of OS/2 LAN Server API."
- Listing all resources and permissions—See "NetAccessEnum" on page 3-15.
- Modifying the current permissions for a resource—See "NetAccessSetInfo" on page 3-25.

NetAccessGetUserPerms

The NetAccessGetUserPerms (partially admin, DOS) function supplies a specified user's or group's permission to a resource. The resource can be a file, directory, drive or logical resource and can be specified remotely by a universal naming convention (UNC) path as well as by a server name.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetAccessGetUserPerms(servername, usergroupname, resource, permission)

char far  *     servername;
char far  *     usergroupname;
char far  *     resource;
short far *     permission;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *usergroupname* points to an ASCIIZ string containing the name of the user or group to be inquired.
- *resource* points to an ASCIIZ string containing the path name for the resource type which can be a directory, file, or drive, as follows:

Type	Format
Directory	*drive:pathname*
File	*drive:pathname*
Pipe	**\pipe***pipename*
Spooler queue	**\print***queuename*
Serial device queue	**\comm***chardevqueue*

- *permission* points to a field where the permission bit field is to be returned.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_BAD_NETPATH	53	The network path cannot be found.

Manifest	Value	Meaning
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ResourceNotFound	2222	The netname cannot be found.
NERR_ACFNotLoaded	2227	The UAS database has not been started.

Manifest	Value	Meaning
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsctl
- DosFsctl(NETTRANSACTION)
- DosFsctl(NULLTRANSACT)
- DosFsctl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosOpen
- DosRead
- DosSemClear.

Remarks

The resource can be specified remotely by a UNC path as well as by a server name. The permissions returned are based on the user's entry and the entry for any groups to which the user belongs. Priority is always given to the user's entry if one exists.

This API requires administrative privilege with the exception that users are always allowed to request their own permissions to any resource. In addition, a user with "P" permission to the resource can get the permissions for any user or group.

NetAccessSetInfo

The NetAccessSetInfo (admin, DOS) function changes an access permission record for a resource.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetAccessSetInfo(servername, resource, level,
                 buf, buflen, parmnum)
char far *       servername;
char far *       resource;
short            level;
char far *       buf;
unsigned short   buflen;
short            parmnum;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *resource* points to an ASCIIZ string containing the local path name for one of the following resource types:

Type	Format
Directory	*drive:pathname*
File	*drive:pathname*
Pipe	**\pipe***pipename*
Spooler queue	**\print***queuename*
Serial device queue	**\comm***chardevqueue*

- *level* specifies the level of detail (1) provided in the *access_info_1* data structure.
- *buf* points to the data structure if *parmnum* is zero. Otherwise, *buf* points to the specific data component that will be changed.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* specifies whether a specific component of the *access_info_1* data structure is being set, or the entire data structure. If *parmnum* is zero, *buf* must contain an *access_info_1* data structure followed by zero or more *access_list* data structures. If *parmnum* is non-zero, only the *acc1_attr* component in the *access_info_1* data structure is set, and *parmnum* must pass the ordinal position value (ACCESS_ATTR_PARMNUM) of the *acc1_attr* component.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_BAD_NETPATH	53	The network path cannot be found.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.

Manifest	Value	Meaning
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ResourceNotFound	2222	The netname cannot be found.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFNoRoom	2228	There are too many names in the access control file.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_ACFTooManyLists	2230	Too many lists were specified.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsctl
- DosFsctl(NETTRANSACTION)
- DosFsctl(NULLTRANSACT)
- DosFsctl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize[-ERROR_DISK_FULL]
- DosOpen

- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite.

Remarks

The specified resource must be a complete path name. For example, file or directory resources must include a drive letter.

A user or application that has ACCESS_PERM permissions for a particular resource can change the access permission record for that resource and remove its permissions for that resource.

The *parmnum* is used only to change the *accl_attr* field in the *access_info_1* data structure. To change the user or group permissions through this API, call NetAccessGetInfo first. Otherwise, if the user or group list is not complete, the information that is not included will be lost.

Related Information

For information on :

- Listing server resources and permissions—See "NetAccessEnum" on page 3-15.
- Retrieving permissions of a resource—See "NetAccessGetInfo" on page 3-18.
- SetInfo functions—See Chapter 1, "Overview of OS/2 LAN Server API."

Alert Category

NetAlertRaise (*local*)—See "NetAlertRaise" on page 3-34.

NetAlertStart (*local*)—See "NetAlertStart" on page 3-37.

NetAlertStop (*local*)—See "NetAlertStop" on page 3-40.

The functions in the Alert category provide a system for notifying network service programs and applications of network events. They are used with the ALERT.H and NETCONS.H include files.

Description

An *event* is a particular instance of a process or state of hardware defined by an application or by the OS/2 LAN Requester/Server software. The OS/2 LAN Requester/Server sends out an *alert*, in the form of a message or the resetting of a semaphore, when certain events occur. Other programs, network services, or internal network components use the NetAlertRaise function to *raise* an alert, notifying various applications or users when a particular type of event occurs.

The ALERT.H include file defines the following classes of events for the alerts that are sent out:

- A print job has completed.
- A user or application received a broadcast message.
- An entry was added to an error log file.
- A network event required administrative assistance.
- A user accessed or used certain applications or resources.

Other classes of alerts can be defined for network applications as needed. For example, if an application routinely writes large amounts of data to a disk drive, running the risk of filling the disk, the user might want the *event* of no free disk space to trigger an alert that notifies the application to pause or end the process slowing the system.

An application or network service, known as a *client*, registers to be notified of an event (or class of events) by calling the NetAlertStart function. Each registration adds an entry to an *alert* table.

A client can receive *alert* messages through one of two delivery mechanisms:

- A mailslot (registered as **\mailslot***name*)
- A system semaphore (registered as **\sem***name*). If a program requires detailed information about an event, it should be registered as a mailslot, since a semaphore cannot transmit such information.

A client can be registered for one type of event or for several types by calling the NetAlertStart function a number of times.

To discontinue alerts for a registered client, use the NetAlertStop function to remove that entry of a client in the *alert* table for the particular class of event.

To change the internal size of the *alert* table (thus allowing more alerts to be defined), a user or application must modify the *numalert* component in the IBMLAN.INI file, and then restart the requester (by using the NET STOP RDR /*y* and NET START RDR commands).

Data Structures

An application registered as a mailslot client receives information about each class of event for which it is registered. This information consists of a fixed-length header followed by variable-length information specific to the type of event, as defined in the ALERT.H include file.

Header Structure

The fixed-length header contains the following data:

```
/* Standard event data structure */

struct std_alert {
  long alrt_timestamp;
  char alrt_eventname[ELVLEN+1];
  char alrt_pad1;
  char alrt_servicename[SNLEN+1];
};
```

where:

- *alrt_timestamp* indicates the time and date of the event.
- *alrt_eventname* is an ASCIIZ string indicating the *alert* class (type of event).
- *alrt_pad1* WORD-aligns data structure components.
- *alrt_servicename* is an ASCIIZ string indicating the application that is raising the alert.

Event Structures

The ALERT.H include file contains data structures for predefined *alert* classes. These structures define only the fixed-length part of the information, not the ASCIIZ strings that follow some of the structures. Each of the six structures is described in the following sections.

Print Request Completed

```
struct print_other_info {
  short alrtpr_jobid;
  short alrtpr_status;
  long  alrtpr_submitted;
  long  alrtpr_size;
};
/* followed by consecutive ASCIIZ strings

char computername[];
char username[];
char queuename[];
char destname[];
char status_string[];

*/
```

where:

- *alrtpr_jobid* is the identification number of the print job.
- *alrtpr_status* indicates the status of the print job.
- *alrtpr_submitted* is a time stamp indicating when the print job was submitted.
- *alrtpr_size* indicates the size (in bytes) of the print job.
- *computername* is an ASCIIZ string indicating the requester or server that submitted the print job.
- *username* is an ASCIIZ string indicating the user that requested the printing.
- *queuename* is an ASCIIZ string indicating the queue that handled the print job.
- *destname* is an ASCIIZ string indicating the printer that handled the job.
- *status_string* is information that the print processor returns. This string corresponds to *status_string* in the *printjob* data structure for the print job.

Network Message Received: In this case, no data structure is defined; however, the text from the received message is in the following format:

```
char msg_text [];
```

where:

- *msg_text* is an ASCIIZ string of message text.

Entry Made to Error Log File

```
struct errlog_other_info {
   short alrter_errcode;
   long alrter_offset;

};
```

where:

- *alrter_errcode* is the error code that was logged.
- *alrter_offset* is the offset for the new entry in the error log file.

Notify Administrator of Network Event

```
struct admin_other_info {
   short alrtad_errcode;
   short alrtad_numstrings;
};

/* followed by 0-9 consecutive ASCIIZ strings

char mergestrings [][];

*/
```

where:

- *alrtad_errcode* is the error code for the new message in the message log file.
- *alrtad_numstrings* indicates the number (0 through 9) of consecutive ASCIIZ strings that *mergestrings* contains.
- *mergestrings* is a series of consecutive ASCIIZ strings that comprise the error message indicated by *alrtad_errcode.*

Notify User of an Event

```
struct user_other_info {
  short alrtus_errcode;
  short alrtus_numstrings;
};

/* followed by the consecutive ASCIIZ strings

char mergestrings [][];
char username[];
char computername[];

*/
```

where:

- *alrtus_errcode* is the error code for the new message in the message log file.
- *alrtus_numstrings* indicates the number (0 through 9) of consecutive ASCIIZ strings that *mergestrings* contains.
- *mergestrings* is a series of consecutive ASCIIZ strings that comprise the error message indicated by *alrtus_errcode.*
- *username* is the user name of the user or application that is being affected by the alert.
- *computername* is the name of the computer that the user or application is accessing.

The ALERT.H include file contains macros to simplify access to the variable-length fields in the *alert* structure as follows:

Macro	Task
ALERT_OTHER_INFO, ALERT_OTHER_INFO_F	When given a pointer to the start of the *std_alert* data structure, the ALERT_OTHER_INFO macro resolves to a pointer to the variable-length part of the *alert* message (the information specific to the *alert* class). Use ALERT_OTHER_INFO_F when a far pointer is required.
ALERT_VAR_DATA, ALERT_VAR_DATA_F	Works with the data structures defined in ALERT.H. Given a pointer to the beginning address of the data structure, ALERT_VAR_DATA returns a pointer to the first variable-length ASCIIZ string. Use ALERT_VAR_DATA_F when a far pointer is required.

Related Information

For information on:

- Include files—See Appendix A, "Include Files."
- Creating mailslots—See "Mailslot Category" on page 3-146.

NetAlertRaise

The NetAlertRaise (local) function notifies all clients registered in the alert table that a particular event has occurred.

Syntax

```
#include <netcons.h>
#include <alert.h>

unsigned far pascal
NetAlertRaise(event, buf, buflen, timeout)
const char far * event;
const char far * buf;
unsigned short   buflen;
unsigned long    timeout;
```

where:

- *event* points to an ASCIIZ string indicating the type of alert to raise. The ALERT.H include file defines the following classes of alerts:

Manifest	ASCIIZ String	Meaning
ALERT_ADMIN_EVENT	"ADMIN"	Notify an administrator.
ALERT_ERRORLOG_EVENT	"ERRORLOG"	Entry added to error log file.
ALERT_MESSAGE_EVENT	"MESSAGE"	User or application received a message.
ALERT_PRINT_EVENT	"PRINTING"	Print job completed or print error.
ALERT_USER_EVENT	"USER"	Application or resource used.

Other classes of events can be defined as necessary.

- *buf* points to the *std_alert* data structure followed by additional alert data.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *timeout* specifies the number of milliseconds to wait for event information to be written to the mailslot.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.

Manifest	Value	Meaning
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_NoSuchAlert	2432	The Alerter service has not been started.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFSRamSemRequest
- DosFreeSeg
- DosFsctl(NETTRANSACTION)
- DosFsctl(NULLTRANSACT)
- DosFsctl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- DosSemRequest.

Remarks

NetAlertRaise notifies all clients registered as semaphores or mailslots when a particular system event takes place. Semaphores are cleared, and mailslots are sent messages.

For mailslot clients, NetAlertRaise writes information from *buf* to clients registered as mailslots by calling the DosWriteMailslot function.

All semaphore clients must be created with the NoExclusive option set and must be called by a process that calls the DosMuxSemWait function on the semaphore. This procedure informs the process of the state transition of the semaphore.

Related Information

For information on:

- Creating a mailslot—See "DosMakeMailslot" on page 3-150.
- Registering a client for an event—See "NetAlertStart" on page 3-37.
- Ending event watching—See "NetAlertStop" on page 3-40.

NetAlertStart

The NetAlertStart (local) function registers a client to be notified of a particular type of network event.

Syntax

```
#include <netcons.h>
#include <alert.h>

unsigned far pascal
NetAlertStart(event, recipient, maxdata)
const char far * event;
const char far * recipient;
unsigned short   maxdata;
```

where:

- *event* points to an ASCIIZ string indicating the type of event of which the client is to be notified. The ALERT.H include file defines the following classes of alerts:

Manifest	ASCIIZ String	Meaning
ALERT_ADMIN_EVENT	"ADMIN"	Notify an administrator.
ALERT_ERRORLOG_EVENT	"ERRORLOG"	Entry added to error log file.
ALERT_MESSAGE_EVENT	"MESSAGE"	User or application received a message.
ALERT_PRINT_EVENT	"PRINTING"	Print job completed or print error.
ALERT_USER_EVENT	"USER"	Application or resource used.

Other classes of events can be defined as necessary.

- *recipient* points to an ASCIIZ string specifying the mailslot or semaphore client to receive the alerts.
- *maxdata* specifies a limit (in bytes) to the information the mailslot client will receive about events in that class.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.

Manifest	Value	Meaning
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_BadEventName	2143	The event name is incorrectly formed.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_AlertExists	2430	The specified client is already registered for the specified event.
NERR_TooManyAlerts	2431	The Alerter service table is full.

Manifest	Value	Meaning
NERR_BadRecipient	2433	The Alerter service recipient is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFSRamSemRequest
- DosFreeSeg
- DosFsctl(NETTRANSACTION)
- DosFsctl(NULLTRANSACT)
- DosFsctl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- DosSemRequest.

Remarks

Event names are ASCIIZ strings stored in ALERT.H. Applications can define their own events, specifying the name when calling the NetAlertStart and NetAlertRaise functions. If you create an event data structure, be sure to choose a name that does not duplicate a name used by another application.

If *recipient* is a semaphore, calling NetAlertRaise for the specified event opens, clears, resets, and closes the system semaphore. The process owning the semaphore must have created it with the NoExclusive option set; presumably such a process will be executing a DosSemWait or DosMuxSemWait function on the semaphore and will have the transition status of the semaphore.

Note that if NetAlertStart starts a particular alert, the alert will still exist (even when a process is ended) until the NetAlertStop function is called to stop the alert.

Related Information

For information on:

- Creating a mailslot—See "DosMakeMailslot" on page 3-150.
- Creating a semaphore—See DosCreateSem in the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference*, Volume 1.
- Reading a mailslot—See "DosReadMailslot" on page 3-152.
- Ending the watch of a client for a class or type of event—See "NetAlertStop" on page 3-40.

NetAlertStop

The NetAlertStop (local) function removes a registered client from the alert table.

Syntax

```
#include <netcons.h>
#include <alert.h>

unsigned far pascal
NetAlertStop(event, recipient)
const char far * event;
const char far * recipient;
```

where:

- *event* points to an ASCIIZ string specifying the class of alerts from which the registered client is to be excluded. The ALERT.H include file defines the following classes of alerts:

Manifest	ASCIIZ String	Meaning
ALERT_ADMIN_EVENT	"ADMIN"	Notify an administrator.
ALERT_ERRORLOG_EVENT	"ERRORLOG"	Entry added to error log file.
ALERT_MESSAGE_EVENT	"MESSAGE"	User or application received a message.
ALERT_PRINT_EVENT	"PRINTING"	Print job completed or print error.
ALERT_USER_EVENT	"USER"	Application or resource used.

Other classes of events can be defined as necessary.

- *recipient* points to an ASCIIZ string containing the user name of the client whose registration is to be canceled.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_NoSuchAlert	2432	The Alerter service has not been started.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFSRamSemRequest
- DosFreeSeg
- DosFsctl(NETTRANSACTION)
- DosFsctl(NULLTRANSACT)
- DosFsctl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- DosSemRequest.

Related Information

For information on registering a client to watch for a particular event, see "NetAlertStart" on page 3-37.

Auditing Category

NetAuditClear (*admin, DOS*)—See "NetAuditClear" on page 3-58.

NetAuditRead (*admin, DOS*)—See "NetAuditRead" on page 3-61.

NetAuditWrite (*local, server*)—See "NetAuditWrite" on page 3-66.

The functions in the Auditing category control the audit log file, which contains an audit trail of operations that occur on a server. They are used with the AUDIT.H and NETCONS.H include files.

Description

Each time a user or application connects to or disconnects from resources on a server, an audit entry can be generated to record the connection or disconnection. Audit entries are stored in an ASCII file. The default audit log file is \IBMLAN\LOGS\NET.AUD. All of the auditing functions perform their operations on this file.

Note: The auditing functions only control changing the contents of the audit log file. To read the audit log file, an application must first call the NetAuditRead function to obtain the handle of the file. The DosRead function can then be called to read the file. To close the file, an application must call the DosClose function.

OS/2 LAN Requester/Server provides for the following types of audit entries:

- A change in a status of a server
- The beginning of a session
- The end of a session
- A password error
- The start of a connection
- A disconnection
- A rejected connection request
- An access made to a resource
- The rejection of an access
- The closing of a file, device, or pipe
- The change of service status code or text
- A modification of access control profile
- A modification of the UAS database
- The logon of a user
- The logoff of a user
- The denial of a logon
- The limit of account exceeded.

Applications can create additional types of audit entries with the NetAuditWrite function.

The other two auditing functions open (NetAuditRead) and clear (NetAuditClear) the audit log file.

Data Structures

All audit entries include a fixed-length header used in conjunction with variable-length data specific to the type of entry. Because of the variable lengths and structures of the *ae_data* portion of the audit entry (it is possible for *ae_data* to be 0 bytes in length), only the fixed header is defined in the *audit_entry* data structure.

The variable-length portion of the audit entry can also contain an offset to a variable-length ASCIIZ string. The offset values are unsigned short integers. To determine the value of the pointer to this string, add the offset value to the address of the *ae_data* structure.

The following example illustrates this procedure. Assume that *ap* points to a buffer containing a complete audit entry and that *ae_type* contains the value AE_CONNSTOP, specifying the predefined *ae_connstop* data structure. To make the variable *computer_name* point to the ASCIIZ string containing the name of the client whose connection was stopped, an application would perform the following algorithm:

```
struct audit_entry * ap;  /* fixed portion of audit entry */
struct ae_connstop * acp; /* variable-length structure */
char * computer_name;     /* pointer to variable-length
                             string */

/* calculate offset to variable-length struct */

acp = (struct ae_connstop *) ((char *) ap + ap-> ae_data_offset);

/* calculate offset to computer name */

computer_name = (char *) acp + acp -> ae_cp_compname;
```

The Fixed-Length Header

The format of the fixed portion of the audit entry is as follows:

```
struct audit_entry {
     unsigned short ae_len;
     unsigned short ae_reserved;
     unsigned long ae_time;
     unsigned short ae_type;
---- unsigned short ae_data_offset; /* offset from
|                                      beginning address of
|                                      audit_entry */
| };
|
| /* variable-length data specific to type of audit entry
|
->   char ae_data[];

     unsigned short ae_len2;

*/
```

where:

- *ae_len* and *ae_len2* specify the length of the audit entry. (Note that *ae_len* is included both at the beginning and the end of the audit entry to enable both backward and forward scanning of the file.) To calculate the entry size, add the size of the *audit_entry* data structure to the size of the variable-length *ae_data* and the size of an unsigned short integer, as follow:

  ```
  totalsize = sizeof (struct audit_entry) +
     sizeof (ae_data) + sizeof (unsigned short);
  ```

- *ae_reserved* is 0.
- *ae_time* is a time stamp indicating the time the audit file log entry was made.
- *ae_type* indicates the type of audit entry. Type values ranging from 0x0000 through 0x07FF are reserved. The NETCONS.H include file defines the following types of data entries:

Manifest	Value	Purpose
AE_SRVSTATUS	0	Status of server changed.
AE_SESSLOGON	1	Session logged on.
AE_SESSLOGOFF	2	Session logged off.
AE_SESSPWERR	3	Password error.
AE_CONNSTART	4	Connection started.
AE_CONNSTOP	5	Connection stopped.
AE_CONNREJ	6	Connection rejected.
AE_RESACCESS2	7	Access granted.
AE_RESACCESSREJ	8	Access rejected.
AE_CLOSEFILE	9	File, device, or pipe closed.
AE_SERVICESTAT	11	Service status code or text changed.

Manifest	Value	Purpose
AE_ACLMOD	12	Access control list modified.
AE_UASMOD	13	User account subsystems database modified.
AE_NETLOGON	14	User logged on to the network.
AE_NETLOGOFF	15	User logged off of the network.
AE_NETLOGDENIED	16	Network logon denied.
AE_ACCLIMITEXCD	17	Account limit exceeded.

- *ae_data_offset* specifies the byte offset from the beginning of the audit entry to the start of the variable-length portion (*ae_data*) of the audit entry. To calculate the start of *ae_data*, add the value of *ae_data_offset* to the starting address of the fixed-length portion of the entry.
- *ae_data* is the variable-length portion of the audit entry, which differs depending on the type of entry specified by *ae_type*. The information starts at *ae_data_offset* bytes from the top of the audit entry. See the following section for information on the structure of each entry type that the OS/2 LAN Requester/Server software defines.

ae_data Structures

The following data structures, specific to the ten types of audit entries, are defined in the AUDIT.H include file. The structures follow (though not necessarily immediately) the *audit_entry* header.

Server Status Changes

```
struct ae_srvstatus {
   unsigned short ae_sv_status;
};
```

where:

- *ae_sv_status* is one of four values indicating a status of the server. These values, defined in AUDIT.H, mean the following:

Manifest	Value	Meaning
AE_SRVSTART	0	Server software started.
AE_SRVPAUSED	1	Server software paused.
AE_SRVCONT	2	Server software restarted.
AE_SRVSTOP	3	Server software stopped.

Session Begins

```
struct ae_sesslogon {
   unsigned short ae_so_compname; /* offset */
   unsigned short ae_so_username; /* offset */
   unsigned short ae_so_privilege;
};
```

where:

- *ae_so_compname* is an offset (from the beginning address of the *ae_sesslogon* data structure) to an ASCIIZ string indicating which requester established the session.
- *ae_so_username* is an offset (from the beginning address of the *ae_sesslogon* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_so_username* and *ae_so_compname* are the same.
- *ae_so_privilege* is one of three values specifying the permission level assigned to *ae_so_username*. These values, defined in AUDIT.H, have the following meanings:

Manifest	Value	Privilege
AE_GUEST	0	Guest
AE_USER	1	User
AE_ADMIN	2	Admin

Session Ends

```
struct ae_sesslogoff {
  unsigned short ae_sf_compname; /* offset */
  unsigned short ae_sf_username; /* offset */
  unsigned short ae_sf_reason;
};
```

where:

- *ae_sf_compname* is an offset (from the beginning address of the *ae_sesslogoff* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_sf_username* is an offset (from the beginning address of the *ae_sesslogoff* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_sf_username* and *ae_sf_compname* are the same.
- *ae_sf_reason* is one of five values indicating why the session was disconnected. These values, defined in AUDIT.H, mean the following:

Manifest	Value	Meaning
AE_NORMAL	0	Normal disconnection or user name limit.
AE_ERROR	1	Error, session disconnect, or bad password.
AE_AUTODIS	2	Auto-disconnect (time out), share removed, or administrative permissions required.
AE_ADMINDIS	3	Administrative disconnection (forced).
AE_ACCRESTRICT	4	Forced off by account system due to account restriction, such as logon hours.

Password Error

```
struct ae_sesspwerr {
  unsigned short ae_sp_compname; /* offset */
  unsigned short ae_sp_username; /* offset */
};
```

where:

- *ae_sp_compname* is an offset (from the beginning of the *ae_sesspwerr* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_sp_username* is an offset (from the beginning of the *ae_sesspwerr* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_sp_username* and *ae_sp_compname* are the same.

Connection Started

```
struct ae_connstart {
  unsigned short ae_ct_compname; /* offset */
  unsigned short ae_ct_username; /* offset */
  unsigned short ae_ct_netname;  /* offset */
  unsigned short ae_ct_connid;
};
```

where:

- *ae_ct_compname* is an offset (from the beginning address of the *ae_connstart* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_ct_username* is an offset (from the beginning address of the *ae_connstart* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_ct_username* and *ae_ct_compname* are the same.
- *ae_ct_netname* is an offset (from the beginning address of the *ae_connstart* data structure) to an ASCIIZ string indicating the netname of the resource with which the connection was made.
- *ae_ct_connid* is the connection identification number.

Connection Stopped

```
struct ae_connstop {
  unsigned short ae_cp_compname; /* offset */
  unsigned short ae_cp_username; /* offset */
  unsigned short ae_cp_netname;  /* offset */
  unsigned short ae_cp_connid;
  unsinged short ae_cp_reason;
};
```

where:

- *ae_cp_compname* is an offset (from the beginning address of the *ae_connstop* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_cp_username* is an offset (from the beginning address of the *ae_connstop* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_cp_username* and *ae_cp_compname* are the same.
- *ae_cp_netname* is an offset (from the beginning address of the *ae_connstop* data structure) to an ASCIIZ string indicating the connected netname of the resource.

- *ae_cp_connid* is the connection identification number.
- *ae_cp_reason* is one of three values indicating why the session was disconnected. These values, defined in AUDIT.H, mean the following:

Manifest	Value	Meaning
AE_NORMAL	0	Normal disconnection, or user name limit.
AE_SESSDIS	1	Error, session disconnect, or bad password.
AE_UNSHARE	2	Autodisconnect (timeout), share removed, or administrative permissions lacking.

Connection Rejected

```
struct ae_connrej {
  unsigned short ae_cr_compname; /* offset */
  unsigned short ae_cr_username; /* offset */
  unsigned short ae_cr_netname;  /* offset */
  unsigned short ae_cr_reason;
};
```

where:

- *ae_cr_compname* is an offset (from the beginning address of the *ae_connrej* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_cr_username* is an offset (from the beginning address of the *ae_connrej* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_cr_username* and *ae_cr_compname* are the same.
- *ae_cr_netname* is an offset (from the beginning address of the *ae_connrej* data structure) to an ASCIIZ string indicating the desired netname of a resource.
- *ae_cr_reason* is one of four values indicating why the session was disconnected. These values are defined in AUDIT.H, as follows:

Manifest	Value	Meaning
AE_USERLIMIT	0	Normal disconnection, or user name limit.
AE_BADPW	1	Error, session disconnect, or bad password.
AE_ADMINPRIVREQD	2	Autodisconnect (timeout), share removed, or administrative permissions lacking.
AE_NOACCESSPERM	3	No access permissions to shared resource.

Access Granted

```
struct ae_resaccess {
  unsigned short ae_ra2_compname; /* offset */
  unsigned short ae_ra2_username; /* offset */
  unsigned short ae_ra2_resname;  /* offset */
  unsigned short ae_ra2_operation;
  unsigned short ae_ra2_returncode;
  unsigned short ae_ra2_restype;
  unsigned short ae_ra2_fileid;
};
```

where:

- *ae_ra2_compname* is an offset (from the beginning address of the *ae_resaccess* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_ra2_username* is an offset (from the beginning address of the *ae_resaccess* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_ra_username* and *ae_ra_compname* are the same.
- *ae_ra2_resname* is an offset (from the beginning address of the *ae_resaccess* data structure) to an ASCIIZ string indicating the name of the resource accessed.
- *ae_ra2_operation* is one of seven values indicating which operation was performed. These values, defined in ACCESS.H, mean the following:

Manifest	Bit Mask	Meaning
ACCESS_READ	0x01	Data was read or executed from a resource.
ACCESS_WRITE	0x02	Data was written to a resource.
ACCESS_CREATE	0x04	An instance of the resource (such as a file) was created; data may have been written to the resource when creating it.
ACCESS_EXEC	0x08	A resource was executed.
ACCESS_DELETE	0x10	A resource was deleted.
ACCESS_ATRIB	0x20	Attributes of a resource were modified.
ACCESS_PERM	0x40	Permissions (read, write, create, execute, and delete) of a resource for a user or application were modified.

- *ae_ra_returncode* gives the return code from the particular operation. If 0, the operation was successful.
- *ae_ra_restype* gives the server message block (SMB) request function code.
- *ae_ra_fileid* gives the server identification number of a file.

Access Rejected

```
struct ae_resaccessrej {
  unsigned short ae_rr_compname; /* offset */
  unsigned short ae_rr_username; /* offset */
  unsigned short ae_rr_resname;  /* offset */
  unsigned short ae_rr_operation;
};
```

where:

- *ae_rr_compname* is an offset (from the beginning address of the *ae_resaccessrej* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_rr_username* is an offset (from the beginning address of the *ae_resaccessrej* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_rr_username* and *ae_rr_compname* are the same.
- *ae_rr_resname* is an offset (from the beginning address of the *ae_resaccessrej* data structure) to an ASCIIZ string indicating the name of the resource to which access was denied.
- *ae_rr_operation* is one of seven values indicating the operation requested. These values are defined in ACCESS.H, as follows:

Manifest	Bit Mask	Meaning
ACCESS_READ	0x01	Data was read or executed from a resource.
ACCESS_WRITE	0x02	Data was written to a resource.
ACCESS_CREATE	0x04	An instance of the resource (such as a file) was created; data may have been written to the resource when creating it.
ACCESS_EXEC	0x08	A resource was executed.
ACCESS_DELETE	0x10	A resource was deleted.
ACCESS_ATRIB	0x20	Attributes of a resource were modified.
ACCESS_PERM	0x40	Permissions (read, write, create, execute, and delete) of a resource for a user or application were modified.

Service Status Code or Text Changed: The audit log entry will be written when service-status auditing is on, and a service performs a NetServiceStatus call that updates the service status (*svcs_status*). Only changes of status to one of the following values cause an audit entry to be written:

- INSTALLED
- UNINSTALLED
- PAUSED

- CONTINUED (ACTIVE).

```
struct ae_servicestat {
  unsigned short ae_ss_compname; /* offset */
  unsigned short ae_ss_username; /* offset */
  unsigned short ae_ss_svcname;  /* offset */
  unsigned short ae_ss_status;
  unsigned long  ae_ss_code;
  unsigned short ae_ss_text;     /* offset */
  unsigned short ae_ss_returnval;
};
```

where:

- *ae_ss_compname* is an offset (from the beginning address of the *ae_servicestat* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_ss_username* is an offset (from the beginning address of the *ae_servicestat* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_ss_username* and *ae_ss_compname* are the same.
- *ae_ss_svcname* is an offset (from the beginning address of the *ae_servicestat* data structure) to an ASCIIZ string indicating the name of a service.
- *ae_ss_status* is the service status being set.
- *ae_ss_code* is the service code being set.
- *ae_ss_text* is an offset to text being set.
- *ae_ss_returnval* is the return value.

Access Control List Modification: The audit log entry will be written when an existing access control list (ACL) record is modified or deleted.

```
struct ae_aclmod {
  unsigned short ae_am_compname; /* offset */
  unsigned short ae_am_username; /* offset */
  unsigned short ae_am_resname;  /* offset */
  unsigned short ae_am_action;
  unsigned long  ae_am_datalen;
};
```

where:

- *ae_am_compname* is an offset (from the beginning address of the *ae_aclmod* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_am_username* is an offset (from the beginning address of the *ae_aclmod* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_am_username* and *ae_am_compname* are the same.
- *ae_am_resname* is an offset (from the beginning address of the *ae_aclmod* data structure) to an ASCIIZ string that indicates the name of a resource that owns the accessed files.
- *ae_am_action* is the action performed on the ACL record, as follows:

Value	Meaning
0	Modification
1	Delete
2	Add

- *ae_am_datalen* is the length of data following the fixed data structure. This is always zero in records generated by the OS/2 LAN Requester/Server.

User Account Subsystem Modification: The audit log entry will be written when an existing user accounts subsystem (UAS) record is modified or deleted, or the UAS modals are modified.

```
struct ae_uasmod {
  unsigned short ae_um_compname; /* offset */
  unsigned short ae_um_username; /* offset */
  unsigned short ae_um_resname;  /* offset */
  unsigned short ae_um_rectype;
  unsigned short ae_um_action;
  unsigned long  ae_um_datalen;
};
```

where:

- *ae_um_compname* is an offset (from the beginning address of the *ae_uasmod* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_um_username* is an offset (from the beginning address of the *ae_uasmod* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_um_username* and *ae_um_compname* are the same.
- *ae_um_resname* is an offset (from the beginning address of the *ae_uasmod* data structure) to an ASCIIZ string indicating the name of a resource that owns the accessed files.
- *ae_um_rectype* is the type of UAS record, as follows:

Value	Meaning
0	User record.
1	Group record.
2	UAS modals.

- *ae_um_action* is the action performed on the UAS record, as follows:

Value	Meaning
0	Modification
1	Deletion
2	Addition

- *ae_um_datalen* is the length of data following the fixed data structure. This is always zero in records generated by the OS/2 LAN Requester/Server.

Network Logon Record: This record is written by the server that processes the network logon of the user.

```
struct ae_netlogon {
  unsigned short ae_no_compname; /* offset */
  unsigned short ae_no_username; /* offset */
  unsigned short ae_no_privilege;
  unsigned short ae_no_authflags;
};
```

where:

- *ae_no_compname* is an offset (from the beginning address of the *ae_netlogon* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_no_username* is an offset (from the beginning address of the *ae_netlogon* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_no_username* and *ae_no_compname* are the same.
- *ae_no_privilege* is the privilege of the user logging on, as follows:

Manifest	Value
AE_GUEST	0
AE_USER	1
AE_ADMIN	2

- *ae_no_authflags* is a reserved field.

Network Logoff Record: This record is written by the server that processes the network logoff of the user.

```
struct ae_netlogoff {
  unsigned short ae_nf_compname; /* offset */
  unsigned short ae_nf_username; /* offset */
  unsigned short ae_nf_reason;
  unsigned short ae_nf_subreason;
};
```

where:

- *ae_nf_compname* is an offset (from the beginning address of the *ae_netlogoff* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_nf_username* is an offset (from the beginning address of the *ae_netlogoff* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_nf_username* and *ae_nf_compname* are the same.
- *ae_nf_reason* is the reason for logoff, as follows:

Manifest	Value	Meaning
AE_NORMAL	0	Normal logoff by user.
AE_ERROR	1	Disconnect due to error.

Manifest	Value	Meaning
AE_AUTODIS	2	Auto-disconnect (station down).
AE_ADMINDIS	3	Administrator disconnected user.
AE_ACCRESTRICT	4	Forced off by account system due to account restriction, such as logon hours.

- *ae_nf_subreason* is the details of reason for logoff. When *nf_reason* is AE_ACCRESTRICT, one of the following is true:

Manifest	Value	Meaning
AE_LIM_UNKNOWN	0	Unknown or unavailable.
AE_LIM_LOGONHOURS	1	Logon hours.
AE_LIM_EXPIRED	2	Account expired.

  Otherwise, this value is zero.

Network Logon Denied: The audit log entry is written when the network logon request is denied.

```
struct ae_netlogdenied {
  unsigned short ae_nd_compname; /* offset */
  unsigned short ae_nd_username; /* offset */
  unsigned short ae_nd_reason;
  unsigned short ae_nd_subreason;
};
```

where:

- *ae_nd_compname* is an offset (from the beginning address of the *ae_netlogdenied* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_nd_username* is an offset (from the beginning address of the *ae_netlogdenied* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_nd_username* and *ae_nd_compname* are the same.
- *ae_nd_reason* is the reason for denial of log on, as follows:

Manifest	Value	Meaning
AE_GENERAL	0	General access denied.
AE_BADPW	1	Incorrect password.
AE_ACCRESTRICT	4	Forced off by account system due to account restriction, such as logon hours.

- *ae_nd_subreason* is the details of reason for denial. When *nd_reason* is AE_ACCRESTRICT, one of the following is true:

Manifest	Value	Meaning
AE_LIM_UNKNOWN	0	Unknown or unavailable.
AE_LIM_LOGONHOURS	1	Logon hours.
AE_LIM_EXPIRED	2	Account expired.
AE_LIM_INVAL_WKSTA	3	Requester ID not valid.
AE_LIM_DISABLED	4	Account disabled.

  Otherwise, this value is zero.

Account Limit Exceeded: The audit log entry is written when users remain logged on while their account limitations no longer permit them to be logged on.

```
struct ae_acclim {
  unsigned short ae_al_compname; /* offset */
  unsigned short ae_al_username; /* offset */
  unsigned short ae_al_resname;  /* offset */
  unsigned short ae_al_limit;
};
```

where:

- *ae_al_compname* is an offset (from the beginning address of the *ae_acclim* data structure) to an ASCIIZ string indicating the requester that established the session.
- *ae_al_username* is an offset (from the beginning address of the *ae_acclim* data structure) to an ASCIIZ string indicating the name of the user who initiated the session. If 0, *ae_al_username* and *ae_al_compname* are the same.
- *ae_al_resname* is the offset to the resource name.
- *ae_al_limit* is the limit that was exceeded, as follows:

Manifest	Value	Meaning
AE_LIM_UNKNOWN	0	Unknown or unavailable.
AE_LIM_LOGONHOURS	1	Logon hours.
AE_LIM_EXPIRED	2	Account expired.

Related Information

For information on:

- OS/2 LAN Requester/Server network commands and IBMLAN.INI file—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- Permission levels—See "User Category" on page 3-382 and "Access Permission Category" on page 3-2.

NetAuditClear

The NetAuditClear (admin, DOS) function clears (and optionally saves) the audit log file of a server.

Syntax

```
#include <netcons.h>
#include <audit.h>

unsigned far pascal
NetAuditClear(servername, backupfile, reserved)
const char far *  servername;
const char far *  backupfile;
char far *        reserved;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *backupfile* points to an ASCIIZ string assigning a name for an optional backup file. The calling application must have Write privileges for the path specified by *backupfile*. The path name must also be accessible by the OS/2 DosMove function. If the path name is relative, it is assumed relative to the IBMLAN\LOGS directory.

  A NULL pointer tells NetAuditClear not to save the audit log entries.
- *reserved* must be NULL.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_PATH_NOT_FOUND	3	The path was not found.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.

Manifest	Value	Meaning
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_FILENAME_EXCED_RANGE	206	The file name is longer than 8 characters or the extension is longer than 3 characters.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CantType	2357	The type of input cannot be determined.

Other error return codes may be returned from the following OS/2 functions:

- DosDelete
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosFsCtlNet.GetRdrAddr()
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosMove
- DosOpen
- DosSemClear
- redir.GetInitPath.

Remarks

NetAuditClear clears the audit log file of all entries and optionally saves the contents to another file.

The NetAuditWrite function (see "NetAuditWrite" on page 3-66) issues an *admin* alert when the audit log file reaches 80% capacity and again when the file reaches 100% capacity. At 100%, NetAuditWrite fails. Therefore, applications should periodically clear the audit log file of outdated information.

To set a maximum size for the audit log file, use one of the following methods:

- Use the NET CONFIG command with the /MAXAUDITLOG option (see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide* for more information).
- Set the *maxauditlog* parameter in the IBMLAN.INI file (see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide* for a description of IBMLAN.INI).
- Call the NetServerSetInfo function with the *sv_maxauditsz* parameter.

Related Information

For information on:

- Getting the status of audit log file capacity—See "NetAlertRaise" on page 3-34.
- Writing an entry to the audit log file—See "NetAuditWrite" on page 3-66.

NetAuditRead

The NetAuditRead (admin, DOS) function opens and returns an OS/2 file handle to the audit log file of a server.

Syntax

```
#include <netcons.h>
#include <audit.h>

unsigned far pascal
NetAuditRead (servername, reserved1, ploghndl, offset, reserved2,
              reserved3, flags, buf, buflen, bytesread, bytesavail)
const char far *            servername;
const char far *            reserved1
HLOG far *                  ploghndl;
unsigned long               offset;
unsigned short far *        reserved2;
unsigned long               reserved3;
unsigned long               flags;
char far *                  buf;
unsigned short              buflen;
unsigned short far *        bytesread;
unsigned short far *        bytesavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *reserved1* must be NULL.
- *ploghndl* is the pointer to the returned log handle.
- *offset* is the record offset to begin read. The offset is ignored unless flags bit 1 is set. If this bit is set, offset is taken as a zero-based offset based on record number, not bytes, at which the data returned should begin. Note that the record offset parameter is zero based from both directions, dependent upon the direction of the read. If reading backwards is specified, then the 0th record is the last record in the file. If reading forward, then the 0th record is the first record in the file.
- *reserved2* must be NULL.
- *reserved3* must be zero.
- *flags* specifies the open flags, bitmapped as shown here.
- *buf* is the pointer to the buffer for returned data.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *bytesread* points to an unsigned short integer indicating the number of bytes that were read into the buffer.
- *bytesavail* points to an unsigned short integer indicating the number of bytes that were available.

The bitmapped *flags* fields are as follows:

Bits	Meaning
0	If 0, the file is read normally. If 1, the file is read backwards and records are returned in the buffer in *reverse-chron* order (newest records first).
1	If 0, read proceeds normally and sequentially. If 1, read proceeds from the Nth record from the start of the file. "N" is the offset parameter.
2-31	Reserved; must be 0.

The offset is ignored unless flags bit 1 is set. If this bit is set, offset is taken as a zero-based offset based on record number, not bytes, at which the data returned should begin.

An application calling NetAuditRead for the first time must initialize the 128-bit log handle as follows:

Bits	Value
127(MSB)-64	0
63-0(LSB)	1

Where the LSB is the last (rightmost) bit. Thereafter, each call to NetAuditRead must be given the value for the log handle that was returned by the previous call to NetAuditRead.

Notes:

1. If *bytesread* is 0 and *bytesavail* is not 0, the buffer is too small to hold the next record in the file.
2. Unlike other uses of *bytesavail*, in this case, the value may be 0xFFFF, which is shorthand for "0xFFFF or more." There can potentially be much more than 64KB data available. The application should continue to process entries until this value is returned to 0.

The data is returned in the buffer. The application should use the *bytesread* value to determine the end of valid data in the buffer.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SHARING_VIOLATION	32	A sharing violation occurred.
ERROR_BAD_NETPATH	53	The network path cannot be found.

Manifest	Value	Meaning
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_NET_WRITE_FAULT	88	A network data fault has occurred.
ERROR_OPEN_FAILED	110	The open/created failed due to explicit fail command.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_FILENAME_EXCED_RANGE	206	The file name is longer than 8 characters or the extension is longer than 3 characters.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.

Manifest	Value	Meaning
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CantType	2357	The type of input cannot be determined.
NERR_LogFileChanged	2378	This log file has changed between reads.
NERR_LogFileCorrupt	2379	This log file is corrupt.
NERR_InvalidLogSeek	2440	The log file does not contain the requested record number.

Other error return codes may be returned from the following OS/2 functions:

- DosChgFilePtr
- DosCLose
- DosDevIOCtl
- DosFSRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosOpen
- DosRead
- DosSemClear
- redir.GetInitPath.

Remarks

After NetAuditRead returns the handle of the audit log file, an application must call the DosRead function to read the contents of the file. To close the file, an application calls the DosClose function.

Related Information

For information on:

- Clearing an audit log file—See "NetAuditClear" on page 3-58.
- Closing an audit log file—See DosClose in *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference*, Volume 1.
- Reading an audit log file—See DosRead in *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference*, Volume 1.
- Writing an entry to the audit log file—See "NetAuditWrite" on page 3-66.

NetAuditWrite

The NetAuditWrite (local, server) function writes an audit trail entry to the local audit log file.

Syntax

```
#include <netcons.h>
#include <audit.h>

unsigned far pascal
NetAuditWrite(type, buf, buflen, reserved1, reserved2)
unsigned short    type;
const char far *  buf;
unsigned short    buflen;
char far *        reserved1;
char far *        reserved2;
```

where:

- *type* specifies the type of entry to write to the file.
- *buf* points to the variable data of the data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *reserved1 and reserved2* are NULL pointers to reserved ASCIIZ strings.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SHARING_VIOLATION	32	A sharing violation occurred.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_BAD_NETPATH	53	The network path cannot be found.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_NET_WRITE_FAULT	88	A network data fault has occurred.
ERROR_OPEN_FAILED	110	The open/created failed due to explicit fail command.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_LogOverflow	2377	This log file exceeds the maximum defined size.

Remarks

NetAuditWrite issues an *admin* alert by calling the NetAlertRaise function when the audit log file reaches 80% capacity and again when the file reaches 100% capacity. At 100% audit log capacity, the NetAuditWrite function fails, returning the error code NERR_LogOverflow.

To return successfully, the NetAuditWrite function requires that the auditing entry in the IBMLAN.INI file be set to YES, as follows:

```
auditing = yes
```

Related Information

For information on:

- Closing the audit log file—See DosClose in *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference*, Volume 1.
- Retrieving size of the audit log file—See "NetServerGetInfo" on page 3-292.
- Setting the maximum size of the audit log file—See "NetServerSetInfo" on page 3-295.

Configuration Category

NetConfigGet2 (*admin, DOS*)—See "NetConfigGet2" on page 3-70.

NetConfigGetAll2 (*admin, DOS*)—See "NetConfigGetAll2" on page 3-73.

The functions in the Configuration category retrieve network configuration information from the IBMLAN.INI file. The NetConfigGet2 function retrieves a single parameter value for a given network component; NetConfigGetAll2 returns all of the parameters for the given component. These functions are used with the CONFIG.H and NETCONS.H include files.

Description

The IBMLAN.INI file is an ASCII file containing the configuration information for OS/2 LAN Requester/Server services. User-defined services and applications also store network configuration information in this file.

The IBMLAN.INI file consists of component lines, parameter lines, and comment lines, in a format that enables the Configuration functions to easily browse through and retrieve the information. The format is as follows:

- Component lines mark the start of information on a component, in the form:

  [*componentname*]

- Parameter lines contain a parameter and a value, in the form:

  parameter = *value*

  The parameter value can consist of arbitrary text and is not processed by the Configuration functions, except that leading and trailing spaces are stripped. Interpretation of the value is left to the caller. No quotation marks are allowed as part of the parameter value.

  For any one component, if a parameter appears several times, NetConfigGetAll2 returns each occurrence; NetConfigGet2 returns only the first instance. The same parameter name can be used under different components without affecting the NetConfigGet2 return.

- Comment lines are any blank lines or lines in which the first nonblank character is a semicolon (;).

  An IBMLAN.INI requester component might contain the following information:

```
[requester]
   ; define net_tool requester
   computername = net_tool
   charcount = 16
```

  As shown, *requester* defines a computer name of *net_tool* and specifies that 16 bytes of characters must accumulate before a requester sends them to a serial device queue.

Note: The IBMLAN.INI file contains default values for network components. These values may not reflect actual values passed to a network service. If you notice inconsistencies between IBMLAN.INI entries and actual service values, examine the manner in which you have defined these service values within your application. To maintain consistency, it is a good idea to call NetConfigGetAll2 and examine the returned values before setting new parameter values for a given component.

DOS Considerations

These APIs cannot be called locally on a DOS requester to retrieve information from the DOSLAN.INI file. However, these APIs can be called remotely on a DOS requester to retrieve information from the IBMLAN.INI file.

Related Information

For information:

- The IBMLAN.INI file—see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- The DOSLAN.INI file—see the *IBM OS/2 DOS LAN Requester User's Guide.*

NetConfigGet2

The NetConfigGet2 (admin, DOS) function retrieves a specified parameter value from the IBMLAN.INI file of a local computer or a remote server.

Syntax

```
#include <netcons.h>
#include <config.h>

unsigned far pascal
NetConfigGet2(servername, reserved, component, parameter,
              buf, buflen, parmlen)
char far *            servername;
char far *            reserved;
char far *            component;
char far *            parameter;
char far *            buf;
unsigned short        buflen;
unsigned short far *  parmlen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local computer.
- *reserved* must be NULL.
- *component* points to an ASCIIZ string specifying the name of the component to be searched.
- *parameter* points to an ASCIIZ string specifying the parameter whose value is to be returned.
- *buf* points to the memory address where the value of a parameter is to be returned.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmlen* points to an unsigned short integer indicating the size (in bytes) of a parameter.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_FILE_NOT_FOUND	2	The file was not found.
ERROR_PATH_NOT_FOUND	3	The path was not found.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SHARING_VIOLATION	32	A sharing violation occurred.

Manifest	Value	Meaning
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_CfgCompNotFound	2146	The program could not find the specified component in the IBMLAN.INI file.
NERR_CfgParamNotFound	2147	The program could not find the specified parameter in the IBMLAN.INI file.

Other error return codes may be returned from the following OS/2 functions:

- DosChgFilePtr
- DosDuphandle
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg
- DosOpen[-ERROR_OPEN_FAILED]
- DosRead
- DosSemClear.

Remarks

NetConfigGet2 returns the value (ASCIIZ string) for a single parameter of a specified component in *buf*. This string is the entire value content of the IBMLAN.INI line for the specified parameter, which is all text to the right of the equal sign (=). Leading and trailing spaces are stripped from this text. No other processing is performed on it.

NetConfigGetAll2

The NetConfigGetAll2 (admin, DOS) function retrieves all configuration information for a given network component in the IBMLAN.INI file of a local computer or a remote server.

Syntax

```
#include <netcons.h>
#include <config.h>

unsigned far pascal
NetConfigGetAll2(servername, reserved, component, buf, buflen,
                 bytesread, bytesavail)
char far *           servername;
char far *           reserved;
char far *           component;
char far *           buf;
unsigned short       buflen;
unsigned short far * bytesread;
unsigned short far * bytesavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local computer.
- *reserved* must be NULL.
- *component* points to an ASCIIZ string specifying the name of the component to search.
- *buf* points to the memory address where the parameter values of a component are to be returned.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *bytesread* points to an unsigned short integer indicating the number of bytes returned to *buf*.
- *bytesavail* points to an unsigned short integer indicating the number of bytes of data that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_FILE_NOT_FOUND	2	The file was not found.
ERROR_PATH_NOT_FOUND	3	The path was not found.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SHARING_VIOLATION	32	A sharing violation occurred.

Manifest	Value	Meaning
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_CfgCompNotFound	2146	The program could not find the specified component in the IBMLAN.INI file.
NERR_LineTooLong	2149	A line in the IBMLAN.INI file is too long.
NERR_JobNotFound	2151	The print job does not exist.

Other error return codes may be returned from the following OS/2 functions:

- DosChgFilePtr
- DosDuphandle
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg
- DosOpen[-ERROR_OPEN_FAILED]
- DosRead
- DosSemClear.

DOS Considerations

This API can be called remotely on a DOS requester to retrieve information from the IBMLAN.INI file.

Remarks

NetConfigGetAll2 returns in *buf* a set of concatenated ASCIIZ strings, representing configuration information for the specified component. Each string is ended by a NULL byte (ASCII 0), and the whole buffer is ended by a NULL string. Information is returned in the form `parm=value`. The parameter name (left of the = sign) is in uppercase. *bytesread* and *bytesavail* are filled in as for GetInfo calls.

For example,

```
"  foo = Bar,1,long comment string "
```

in the IBMLAN.INI file is returned as:

```
"FOO=Bar,1,long comment string"
```

Connection Category

NetConnectionEnum (admin, server, DOS)—See "NetConnectionEnum" on page 3-78.

The NetConnectionEnum function lists all connections made to a server by a requester client or all connections made to a shared resource of a server. The function is used with the SHARES.H and NETCONS.H include files.

Description

A requester accesses a shared resource of a server by means of a connection. Thus, a *connection* is the path between a redirected local device name of a requester and a shared resource of a server. Using a NetUseAdd (UNC) name can establish a connection without any local device name.

Data Structures

The NetConnectionEnum function returns data at a detail level of 0 or 1, using the following data structures:

Connection Information (Level 0)

```
struct connection_info_0 {
  unsigned short coni0_id;
};
```

where:

- *coni0_id* is the connection identification number.

Connection Information (Level 1)

```
struct connection_info_1 {
  unsigned short coni1_id;
  unsigned short coni1_type;
  unsigned short coni1_num_opens;
  unsigned short coni1_num_users;
  unsigned long  coni1_time;
  char far *     coni1_username;
  char far *     coni1_netname;
};
```

where:

- *coni1_id* is the connection identification number.
- *coni1_type* indicates the type of connection made from the local device name to the shared resource. The SHARES.H include file defines the following types of connection:

Manifest	Value	Meaning
STYPE_DISKTREE	0	Disk connection.
STYPE_PRINTQ	1	Spooler queue connection.
STYPE_DEVICE	2	Serial device connection.
STYPE_IPC	3	Interprocess communication (IPC) connection.

- *conil_num_opens* indicates the number of files that are currently open as a result of the connection.
- *conil_num_users* indicates the number of users on the connection.
- *conil_time* indicates the number of seconds the connection has been established.
- *conil_username* points to an ASCIIZ string indicating the user that made the connection.
- *conil_netname* points to an ASCIIZ string indicating either the netname of the shared resource of the server or the computer name of the requester, depending on which name was specified as the *qualifier* parameter of the NetConnectionEnum function. The type of name supplied to *conil_netname* is the inverse of the type supplied to the *qualifier* parameter.

Related Information

For information on connecting a device name of a requester to a shared resource of a server, see "NetUseAdd" on page 3-372.

NetConnectionEnum

The NetConnectionEnum (admin, server, DOS) function gives a listing of connections made to a shared resource of a server, or of all connections established from a particular computer to a server.

Syntax

```
#include <netcons.h>
#include <shares.h>

unsigned far pascal
NetConnectionEnum(servername, qualifier, level, buf,
                  buflen, entriesread, totalentries)
const char far *     servername;
const char far *     qualifier;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far * entriesread;
unsigned short far * totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local server.
- *qualifier* points to an ASCIIZ string specifying either the netname of the shared resource whose connections will be listed or the client name of the requester whose connections to the shared resource will be listed (*qualifier* cannot be a NULL pointer or string).
- *level* specifies the level of detail (0 or 1) for the returned *connection_info* data.
- *buf* points to the *connection_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.

Manifest	Value	Meaning
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_NetNameNotFound	2310	This shared resource does not exist.
NERR_ClientNameNotFound	2312	A session does not exist with that computer name.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

If *qualifier* specifies a requester, NetConnectionEnum returns a list of all connections made between the requester and the specified server during the current session.

When *qualifier* specifies a shared resource, NetConnectionEnum returns a list of all connections made to the shared resource.

Related Information

For information on:

- Listing all available servers—See "NetServerEnum2" on page 3-289.
- Listing sessions on a server—See "NetSessionEnum" on page 3-331.

Domain Category

NetGetDCName (*DOS*)—See "NetGetDCName" on page 3-82.

NetLogonEnum (*partially admin, DOS*)—See "NetLogonEnum" on page 3-85.

The functions in the Domain category deal specifically with the information of a domain and are exclusive of other categories. They are used with the ACCESS.H and NETCONS.H include files.

Description

The functions in this category deal with domain-specific information. The NetGetDCName function obtains the name of the domain controller when provided the name of the domain. The NetLogonEnum function enumerates the information of logged-on users in a domain. The information is in the level 0, 1, or 2 *user_logon_info* data structures. See the data structure information under "User Category" on page 3-382.

NetGetDCName

Given a domain name, the NetGetDCName (DOS) function returns the name of the domain controller if there is any. The NULL domain name is taken to mean obtain the domain controller (DC) of the primary domain.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGetDCName(servername, domain, buf, buflen)
char far *      servername;
char far *      domain;
char far *      buf;
unsigned short  buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *domain* points to an ASCIIZ string containing the name of the domain.
- *buf* points to the buffer for the name of the domain controller to be returned.
- *buflen* specifies the size (in bytes) of the buffer *buf*.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_DCNotFound	2453	No domain controller was found on this domain.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosDeleteMailslot
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg
- DosMakeMailslot[-ERROR_ALREADY_EXISTS]
- DosSemClear.

Remarks

If the return code is 0 (success), the buffer contains an ASCIIZ string representing the name of the domain controller as a UNC name, for example, "\\server."

Because NetGetDCName attempts to find the domain controller for the specified domain each time it is called, this function may affect the performance of applications that call it often. However, since the domain controller of a domain may change, applications should not cache the domain controller name for more than a small set of operations. It is recommended that the application be written without more than very local caching, unless performance tests indicate that calls to NetGetDCName are the specific cause of poor performance.

Even then, the application should take care to refresh its internal cache by calling this function when possible.

NetLogonEnum

The NetLogonEnum (DOS) function supplies information about logged-on users.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetLogonEnum(servername, level, buf,
             buflen, entriesread, totalentries)
char far *           servername;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far * entriesread;
unsigned short far * totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level (0 or 2) of detail supplied to the data structure.
- *buf* points to the *user_logon_info* data structures.
- *buflen* specifies the size (in bytes) of the *user_logon_info* data structure.
- *entriesread* contains the number of entries on return.
- *totalentries* contains the total entries available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_InvalidUASOp	2451	This operation is not permitted when the Netlogon service is running.
NERR_NetLogonNotStarted	2455	The Netlogon service has not been started.

Manifest	Value	Meaning
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize[-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite.

Error Logging Category

NetErrorLogClear (*admin*)—See "NetErrorLogClear" on page 3-90.

NetErrorLogRead (*admin*)—See "NetErrorLogRead" on page 3-93.

NetErrorLogWrite (*local*)—See "NetErrorLogWrite" on page 3-97.

The functions in the Error Logging category control the error log file. They are used with the ERRLOG.H and NETCONS.H include files.

Description

Each time an error condition occurs during a network operation, an error log entry can be generated by NetErrorLogWrite to record error information. The other two functions enable opening (NetErrorLogRead) and clearing (NetErrorLogClear) of the error log file (which stores the entries).

Error log entries are stored as ASCII text. The default error log file name is \IBMLAN\LOGS\NET.ERR. All error logging functions perform their operations on this file.

Note: The error logging functions control changes to the error log file only. To read the error log file, an application must first call the NetErrorLogRead function to obtain the handle of the file. The DosRead function can then be called to read the file. To close the file, an application must call the DosClose function.

The error log file contains information about the following types of errors:

- OS/2 LAN Requester/Server software internal errors
- OS/2 internal errors
- Network service errors.

Data Structures

The NetErrorLogWrite function uses the *error_log* data structure to write an entry to the error log file. The entry consists of a fixed-length data structure optionally followed by zero or more ASCIIZ strings (*el_text*) describing the error message and a block of raw data (*el_data*) relating to the cause of the error. Because of the variable lengths and structures of the *el_data* and *el_text* portions of the entry, only the fixed-length data structure is defined in the *error_log* data structure.

The fixed portion of the error log entry has the following format:

```
struct error_log {
        unsigned short el_len;
        unsigned short el_reserved;
        unsigned long  el_time;
        unsigned short el_error;
        char           el_name[SNLEN+1];
-----   unsigned short el_data_offset; /* offset from beginning
|                                                 address of error_log */
|       unsigned short el_nstrings;
| };
|
|    /* variable-length data specific to the error
|       message and block of data associated with error  */
|
|       char el_text []; /* error message */
-->     char el_data []; /* raw data - the number of bytes
                                 used for raw data is equivalent to:
                                 size = el_len - (el_data_offset
                                        + sizeof(el_len) ); */
        unsigned short el_len;
```

where:

- *el_len* indicates the length (in bytes) of the error log entry. (Note that *el_len* is included at both the beginning and end of the entry to enable both forward and backward scanning of the file.)
- *el_reserved* is reserved.
- *el_time* indicates the time when *el_name* submitted the error entry.
- *el_error* is the error code for the error. *el_error* can be used to obtain an error message from the NET.MSG file.
- *el_name* is an ASCIIZ string indicating the name of the network service or application that returned the error entry.
- *el_data_offset* specifies the byte offset from the beginning of the error log entry to the start of its variable-length portion (*el_data*).
- *el_nstrings* indicates the number of ASCIIZ strings the *el_text* portion of the entry contains.
- *el_text* points to zero or more ASCIIZ strings describing the error.
- *el_data* points to the raw data associated with the error.

Related Information

For information on error codes, see Appendix C, "Return Codes."

NetErrorLogClear

The NetErrorLogClear (admin) function clears (and optionally saves) the error log file of a computer.

Syntax

```
#include <netcons.h>
#include <errlog.h>

unsigned far pascal
NetErrorLogClear(servername, backupfile, reserved)
const char far * servername;
const char far * backupfile;
char far *       reserved;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *backupfile* points to an ASCIIZ string assigning a name for an optional backup file. The calling application must have Write privileges for the path specified by *backupfile*. The path name must also be accessible by the OS/2 DosMove function. If the path name is relative, it is assumed relative to the IBMLAN\LOGS directory.

  A NULL pointer indicates that NetErrorLogClear is not to save the error log entries.
- *reserved* is a NULL pointer.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SHARING_VIOLATION	32	A sharing violation occurred.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_BAD_NETPATH	53	The network path cannot be found.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_BAD_NET_NAME	67	This network name cannot be found.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.

Manifest	Value	Meaning
ERROR_NET_WRITE_FAULT	88	A network data fault has occurred.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_FILENAME_EXCED_RANGE	206	The file name is longer than 8 characters or the extension is longer than 3 characters.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CantType	2357	The type of input cannot be determined.

Other error return codes may be returned from the following OS/2 functions:

- DosDelete
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize[-ERROR_DISK_FULL]
- DosMove
- DosOpen
- DosSemClear
- redir.GetNetInitPath.

Remarks

NetErrorLogClear fails if the error log file is currently opened by another process. The NetErrorLogWrite function (see "NetErrorLogWrite" on page 3-97) issues an *admin* alert when the error log file reaches 80% capacity and again when the file reaches 100% capacity. At 100% error log file capacity, NetErrorLogWrite fails. Therefore, applications should periodically clear the error log file of outdated information.

To set a maximum size for the error log file, use one of the following methods:

- Use the NET CONFIG command with the /MAXERRORLOG option (see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide* for more information).
- Set the *maxerrorlog* parameter in the IBMLAN.INI file (see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide* for a description of the IBMLAN.INI file).
- Call the NetWkstaSetInfo function with the *wki0_errlogsz* parameter.

Related Information

For information on writing an entry to the error log file, see "NetErrorLogWrite" on page 3-97.

NetErrorLogRead

The NetErrorLogRead (admin) function opens and returns an OS/2 file handle to the error log file of a computer .

Syntax

```
#include <netcons.h>
#include <errlog.h>

unsigned far pascal
NetErrorLogRead (servername, reserved1, ploghndl, offset, reserved2,
                 reserved3, flags, buf, buflen, bytesread, bytesavail)
const char far *            servername;
const char far *            reserved1
HLOG far *                  ploghndl;
unsigned long               offset;
unsigned short far *        reserved2;
unsigned long               reserved3;
unsigned long               flags;
char far *                  buf;
unsigned short              buflen;
unsigned short far *        bytesread;
unsigned short far *        bytesavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local server.
- *reserved1* must be a NULL pointer.
- *ploghndl* is the pointer to the returned log handle.
- *offset* is the record offset to begin read. The *offset* is ignored unless flags bit 1 is set. If this bit is set, offset is taken as a zero-based offset based on record number rather than bytes, at which the data returned should begin. Note that the record offset parameter is zero based from both directions, dependent upon the direction of the read. If reading backwards is specified, then the 0th record is the last record in the file. If reading forward, then the 0th record is the first record in the file.
- *reserved2* must be a NULL pointer.
- *reserved3* must be zero.
- *flags* specifies the open flags, bitmapped as shown here.
- *buf* is the pointer to the buffer for returned data.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *bytesread* points to an unsigned short integer indicating the number of bytes read into the buffer.
- *bytesavail* points to an unsigned short integer indicating the number of bytes available.

The bitmapped *flags* fields are as follows:

Bits	Meaning
0	If 0, the file is read normally. If 1, the file is read backwards and records are returned in the buffer in *reverse-chron* order (newest records first).
1	If 0, read proceeds normally and sequentially. If 1, read proceeds from the Nth record from the start of the file. "N" is the offset parameter.
2-31	Reserved; must be 0.

The offset is ignored unless flags bit 1 is set. If this bit is set, offset is taken as a zero-based offset based on record number, not bytes, at which the data returned should begin.

An application calling NetErrorLogRead for the first time must initialize the 64-bit log handle as follows:

Bits	Value
127 (MSB)-64	0
63-0(LSB)	1

Where the least significant bit (LSB) is the last (rightmost) bit. Thereafter, each call to NetErrorLogRead must be given the value for the log handle that was returned by the previous call to NetErrorLogRead.

Note: If *bytesread* is 0 and *bytesavail* is not 0, the buffer is too small to hold the next record in the file.

Unlike other API uses of *bytesavail*, in this case, the value may be 0xFFFF, which is shorthand for "0xFFFF or more." There can potentially be much more than 64KB of data available. The application should continue to process entries until this value is returned to 0.

The data is returned in the buffer. The application should use the *bytesread* value to determine the end of valid data in the buffer.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_BAD_NETPATH	53	The network path cannot be found.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.

Manifest	Value	Meaning
ERROR_BAD_NET_NAME	67	This network name cannot be found.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_NET_WRITE_FAULT	88	A network data fault has occurred.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_FILENAME_EXCED_RANGE	206	The file name is longer than 8 characters or the extension is longer than 3 characters.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CantType	2357	The type of input cannot be determined.

Manifest	Value	Meaning
NERR_LogFileChanged	2378	This log file has changed between reads.
NERR_LogFileCorrupt	2379	This log file is corrupt.
NERR_InvalidLogSeek	2440	The log file does not contain the requested record number.

Other error return codes may be returned from the following OS/2 functions:

- DosChgFilePtr
- DosClose
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosOpen
- DosRead
- DosSemClear
- redir.GetNetInitPath.

Remarks

After the NetErrorLogRead function returns the handle of the error log file , an application calls the DosRead function to read the contents of a file. To close the file, an application must call the DosClose function.

Related Information

For information on:

- Clearing an error log file—See "NetErrorLogClear" on page 3-90.
- Closing an error log file—See DosClose in the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference.*
- Reading an error log file—See DosRead in the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference.*

NetErrorLogWrite

The NetErrorLogWrite (local) function writes an entry to the error log file of a computer.

Syntax

```
#include <netcons.h>
#include <errlog.h>

unsigned far pascal
NetErrorLogWrite(reserved1, code, component, buf, buflen,
                 insbuf, nstrings, reserved2);
char far *       reserved1;
unsigned short   code;
const char far * component;
const char far * buf;
unsigned short   buflen;
const char far * insbuf;
unsigned short   nstrings;
char far *       reserved2;
```

where:

- *reserved1* must be a NULL pointer.
- *code* specifies the error code of the network error that occurred.
- *component* points to an ASCIIZ string specifying which software component encountered the error.
- *buf* points to the raw data associated with the error condition.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *insbuf* points to the ASCIIZ strings containing the error message.
- *nstrings* indicates the number of concatenated ASCIIZ strings *insbuf* stores.
- *reserved2* must be a NULL pointer.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_LogonNoUserPath	2211	The server is configured without a valid user path.

Manifest	Value	Meaning
NERR_LogOverflow	2377	This log file exceeds the maximum defined size.

Remarks

The NetErrorLogWrite function internally calls the appropriate OS/2 functions to open and close the error log file.

The NetErrorLogWrite function issues an error log alert (with NetAlertRaise) each time an entry is written to the error log file. Also, the NetErrorLogWrite function issues an *admin* alert by calling the NetAlertRaise function when the error log file reaches 80% capacity and again when the file reaches 100% capacity. At 100% error log file capacity, NetErrorLogWrite fails, returning the error code NERR_LogOverflow.

Related Information

For information on:

- Clearing the error log file—See "NetErrorLogClear" on page 3-90.
- Closing the error log file—See DosClose in *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference.*
- Error codes—See Appendix C, "Return Codes."
- Limiting the size of the error log file—See "NetServerSetInfo" on page 3-295.
- Retrieving the size of the error log file—See "NetServerGetInfo" on page 3-292.

File Category

NetFileClose2 (*admin, server, DOS*)—See "NetFileClose2" on page 3-101.

NetFileEnum2 (*admin, server, DOS*)—See "NetFileEnum2" on page 3-104.

NetFileGetInfo2 (*admin, server, DOS*)—See "NetFileGetInfo2" on page 3-107.

The functions in the File category provide a system for monitoring which file, device, and pipe resources are opened on a server and for closing one of these resources if necessary. They are used with the SHARES.H and NETCONS.H include files.

Description

NetFileGetInfo2 returns information on one particular opening of a resource. Two levels of detail are available, yielding only the identification number assigned to the resource when it was opened (level 2) or additional data on permissions, file-locks, and who opened the resource (level 3).

NetFileClose2 forces a resource closed when a system error prevents normal closure by the DosClose function.

Data Structures

The *level* parameter for NetFileEnum2 and NetFileGetInfo2 specifies one of two levels of information (2 or 3) to be returned. Both functions return data structured as follows:

Opened Resources (Level 2)

```
struct file_info_2 {
unsigned long  fi2_id;
};
```

where:

- *fi2_id* is the identification number assigned to the resource at opening.

Opened Resources (Level 3)

```
struct file_info_3 {
   unsigned long  fi3_id;
   unsigned short fi3_permissions;
   unsigned short fi3_num_locks;
   char far *     fi3_pathname;
   char far *     fi3_username;
};
```

where:

- *fi3_id* is the identification number assigned to the resource at opening.

- *fi3_permissions* indicates the access permissions of the opening application. The bit mask of *fi3_permissions* is defined in SHARES.H as follows:

Manifest	Bitmask	Meaning
FILE_READ	0x1	Permission to read a resource, and by default, execute the resource.
FILE_WRITE	0x2	Permission to write to a resource.
FILE_CREATE	0x4	Permission to create a resource; data can be written when creating the resource.

- *fi3_num_locks* indicates the number of file-locks on the file, device, or pipe.
- *fi3_pathname* points to an ASCIIZ string giving the path name of the opened resource.
- *fi3_username* points to an ASCIIZ string indicating the user that opened the resource.

NetFileClose2

The NetFileClose2 (admin, server, DOS) function forces a resource closed when a system error prevents a normal DosClose function closing.

Syntax

```
#include <netcons.h>
#include <shares.h>

unsigned far pascal
NetFileClose2(servername, fileid)
const char far * servername;
unsigned long   fileid;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *fileid* is the identification number assigned to the resource at opening.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.

Manifest	Value	Meaning
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_FileIdNotFound	2314	There is not an open file with that ID number.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_NoSuchServer	2460	The server ID is not valid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

Normally, the DosClose function is used to close a resource opened by a call to the DosOpen function. Use NetFileClose2 to force closed a resource opened by another process.

Related Information

For information on listing all open files and their identification numbers for a server, see "NetFileEnum2" on page 3-104.

NetFileEnum2

The NetFileEnum2 (admin, server, DOS) function supplies information about some or all open files on the server, allowing the user to supply a key to get the required information through iterated calls to the API.

Syntax

```
#include <netcons.h>
#include <shares.h>

unsigned far pascal
NetFileEnum2( servername, basepath, username, level, flags, buf,
              buflen, entriesread, totalentries, resume_key )
char far *                servername;
char far *                basepath;
char far *                username;
short                     level;
char far *                buf;
unsigned short            buflen;
unsigned short far *      entriesread;
unsigned short far *      totalentries;
void far *                resume_key;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *basepath* is the base path for enumeration. If non-NULL, *basepath* serves as a qualifier to the enumeration. The entries returned are limited to those whose name begins with the qualifier string. For example, a *basepath* of C:TMP would enumerate only open files whose *pathnames* begin with C:TMP, including C:TMPFILE and C:\TMP\DOCUMENT.
- *username* points to an ASCIIZ string indicating the name of the user. If non-NULL, *username* serves as a qualifier to the enumeration. The files returned are limited to those whose opener *username* matches the qualifier.
- *level* specifies the level of detail (2 or 3) in the *file_info_* data structure.
- *buf* points to the *file_info_2* or *file_info_3* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.
- *resume_key* is a pointer to structure FRK (structure *res_file_enum2*). This field is used for continuing scanning.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ServerNotStarted	2114	The Server service has not been started.

Remarks

This API provides a way for the user to overcome the problem arising when the information returned exceeds 64KB. To initialize the key *resume_key*, use the macroinstruction FRK_INIT supplied in the file SHARES.H, which accepts a structure FRK as an argument. The following is an example of an application code segment:

```
FRK f;

FRK_INIT ( f );
NetFileEnum2 ( ..., &f, ... );
```

When invoked with an initial resume key, if the supplied buffer is too small to return all the requested information, the NetFileEnum2 function returns the error code ERROR_MORE_DATA and a *resume_key* suitable for retrieving the remaining data. When invoked with a *resume_key* from a previous call, it resumes the enumeration where indicated by **resume_key* The user must not attempt to set this key other than to initialize it. Other values of **resume_key* supplied by the user must have been returned by a preceeding call to NetFileEnum2.

NetFileEnum2 never returns an entry that has partial data; that is, a fixed-length data record and all variable-length data is present for each returned item. Items that cannot fit completely are not returned in the buffer. This differs from normal Enum function calls, which return partial data for some entries, usually the last few, if the buffer is too small. The reason that Enum2 differs is because the entries can be retrieved in full by subsequent calls (using the *resume_key*), and so partial data could be misleading and is less useful than in normal Enum functions.

The *username* parameter, if not NULL, serves as a qualifier to the enumeration. The files returned are limited to those whose opener user name matches the qualifier.

The *basepath* parameter, if not NULL, serves as a prefix to qualify the enumeration. The entries returned are limited to those whose names begin with the qualifier string.

For example, a basepath of "C:\TMP" would enumerate only open files whose path names begin with "C:\TMP," including "C:\TMPFILE" and "C:\TMP\DOCUMENT."

If both the *username* and the *basepath* parameters are specified, only the files matching both the qualifying conditions are returned.

NetFileGetInfo2

The NetFileGetInfo2 (admin, server, DOS) function retrieves information about a particular opening of a server resource.

Syntax

```
#include <netcons.h>
#include <shares.h>

unsigned far pascal
NetFileGetInfo2(servername, fileid, level,
              buf, buflen, totalavail)
const char far *      servername;
unsigned long        fileid;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *fileid* indicates the identification number assigned to the resource at opening.
- *level* specifies the level of detail (2 or 3) to be returned by the *file_info* data structure.
- *buf* points to the *file_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_FileIdNotFound	2314	There is not an open file with that ID number.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFSRamSemClear
- DosFreeSeg
- DosFsCtl
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Related Information

For information on:

- Closing a file, device, or pipe—See "NetFileClose2" on page 3-101.
- Listing files, devices, or pipes open on a server—See "NetFileEnum2" on page 3-104.

Group Category

NetGroupAdd (*admin, DOS*)—See "NetGroupAdd" on page 3-112.

NetGroupAddUser (*admin, DOS*)—See "NetGroupAddUser" on page 3-115.

NetGroupDel (*admin, DOS*)—See "NetGroupDel" on page 3-118.

NetGroupDelUser (*admin, DOS*)—See "NetGroupDelUser" on page 3-121.

NetGroupEnum (*partially admin, DOS*)—See "NetGroupEnum" on page 3-124.

NetGroupGetInfo (*partially admin, DOS*)—See "NetGroupGetInfo" on page 3-127.

NetGroupGetUsers (*partially admin, DOS*)—See "NetGroupGetUsers" on page 3-130.

NetGroupSetInfo (*admin, DOS*)—See "NetGroupSetInfo" on page 3-133.

NetGroupSetUsers (*admin, DOS*)—See "NetGroupSetUsers" on page 3-136.

The functions in the Group category control user groups in the user accounts subsystem (UAS) database. They are used with the ACCESS.H and NETCONS.H include files.

Description

A *group* is a set of users sharing common permissions in the UAS database. The Group functions create or delete groups and review or adjust their membership.

Access permissions can be assigned for all members of a group by supplying the group name to the NetAccessAdd function (see "Access Permission Category" on page 3-2) instead of individually assigning each user an access permission record.

Note: The OS/2 LAN Requester/Server software maintains special groups to which any user assigned USER or ADMIN privileges is added automatically. If an application calls any of the Group functions in an attempt to modify the group USERS, the group ADMIN, or their membership, the function returns the NERR_SpeGroupOp error code.

To create a user group, an application calls the NetGroupAdd function, supplying a group name. Initially, the group has no members. Members are assigned to the group by calling NetGroupAddUser.

NetGroupDelUser removes the name of a specified user from a group, and NetGroupDel disbands a group. (NetGroupDel works regardless of whether or not the group has members.)

Two functions retrieve information about groups on a server. NetGroupEnum produces a list of all groups. NetGroupGetUsers lists all members of a specified group.

Special Groups

There are three special groups: USERS, ADMINS, and GUESTS. Each user account automatically belongs to one of these three special groups according to the user's privilege level. The members of these special groups must have one of the following privilege levels:

- USER_PRIV_USER
- USER_PRIV_ADMIN
- USER_PRIV_GUEST.

Users cannot be deleted from these groups, nor can groups be deleted. An attempt to delete groups or users in these groups causes the NERR_SpeGroupOp error code to be returned.

Data Structures

Only three of the Group functions—NetGroupAdd, NetGroupEnum, and NetGroupGetUsers—return structured data. The simple data structures that these functions use are described following the syntax description for each function.

Group Information (Level 0 and Level 1)

The basic data structures for Group information are as follows:

```
struct group_info_0 {
   char        grpi0_name[GNLEN+1];
};
```

where:

- *grpi0_name* is the name of the group.

```
struct group_info_1 {
   char        grpi1_name[GNLEN+1];
   char        grpi1_pad_1;
   char far *  grpi1_comment;
};
```

where:

- *grpi1_name* is the name of the group.
- *grpi1_pad_1* is for the WORD-alignment in the data structure.
- *grpi1_comment* points to an ASCIIZ string containing the comment or remark of the group. The string can be NULL.

Group Membership Information (Level 0)

The basic data structure for Group Membership information is as follows:

```
struct group_users_info_0 {
   char        grui0_name[UNLEN+1];
};
```

where:

- *grpi0_name* is the name of the user in the group.

All of these functions should be used with the ACCESS.H and NETCONS.H include files.

NetGroupAdd

The NetGroupAdd (admin, DOS) function creates a new group account in the user accounts subsystem (UAS) database.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGroupAdd (servername, level, buf, buflen)
char far *     servername;
short          level;
char far *     buf;
unsigned short buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (0 or 1) for the *group_info* data structure.
- *buf* points to the *group_info* data structure.

  When adding at level 0, the comment field is set to the empty string, since no comment field is provided in the level 0 structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_BAD_NETPATH	53	The network path cannot be found.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_BadUsername	2202	The user name or group name parameter is invalid.
NERR_GroupExists	2223	The group name is already in use.
NERR_UserExists	2224	The user account already exists.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFNoRoom	2228	There are too many names in the access control file.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.

Manifest	Value	Meaning
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

NetGroupAdd will fail if the name already is used as a user name. User names and group names must be unique.

Related Information

For information on:

- Adding a user to a group—See "NetGroupAddUser" on page 3-115.
- Assigning group permissions—See "NetAccessAdd" on page 3-6.
- Deleting a group account from a server—See "NetGroupDel" on page 3-118.
- Listing all groups on a server—See "NetGroupEnum" on page 3-124.

NetGroupAddUser

The NetGroupAddUser (admin, DOS) function adds a user to a group in the user accounts subsystem (UAS) database.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGroupAddUser(servername, groupname, username)
char far * servername;
char far * groupname;
char far * username;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local server.
- *groupname* points to an ASCIIZ string specifying the group the user will join.
- *username* points to an ASCIIZ string specifying the user to add to the group.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.

Manifest	Value	Meaning
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_GroupNotFound	2220	The group does not exist.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_SpeGroupOp	2234	This operation is not allowed on this special group.
NERR_UserInGroup	2236	The user already belongs to this group.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize[-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite.

Remarks

If any attempt is made to add a user name to the special groups (USERS, ADMINS, or GUESTS), the NetGroupAddUser function returns the NERR_SpeGroupOp error code.

Related Information

For information on:

- Creating a new group—See "NetGroupAdd" on page 3-112.
- Defining group access permission records—See "Access Permission Category" on page 3-2.
- Removing a user from a group—See "NetGroupDelUser" on page 3-121.
- Retrieving a list of the members of a group—See "NetGroupGetUsers" on page 3-130.
- Setting the groups of which a user is a member—See "NetUserSetGroups" on page 3-423.

NetGroupDel

The NetGroupDel (admin, DOS) function removes a group account from the user accounts subsystem (UAS) database.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGroupDel(servername, groupname)
char far * servername;
char far * groupname;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *groupname* points to an ASCIIZ string specifying which group to remove.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_FILENAME_EXCED_RANGE	206	The file name is longer than 8 characters or the extension is longer than 3 characters.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.

Manifest	Value	Meaning
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_GroupNotFound	2220	The group does not exist.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_SpeGroupOp	2234	This operation is not allowed on this special group.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize[-ERROR_DISK_FULL]
- DosSemClear
- DosWrite.

Remarks

It is not necessary to remove all members from a group before deleting the group account.

Deleting a group account does not delete the individual accounts of its member users.

Deleting a group deletes it from the access control profiles.

NetGroupDel returns the NERR_SpeGroupOp error code if any attempt is made to remove the special groups (USERS, ADMINS, or GUESTS).

Related Information

For information on:

- Adding a group to the UAS database—See "NetGroupAdd" on page 3-112.
- Listing all groups in the UAS database—See "NetGroupEnum" on page 3-124.
- Removing a user from a group—See "NetGroupDelUser" on page 3-121.
- Retrieving a list of members for a group—See "NetGroupGetUsers" on page 3-130.

NetGroupDelUser

The NetGroupDelUser (admin, DOS) function removes a user from a particular group in the user accounts subsystem (UAS) database.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGroupDelUser(servername, groupname, username)
char far * servername;
char far * groupname;
char far * username;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local server.
- *groupname* points to an ASCIIZ string specifying the group to be altered.
- *username* points to an ASCIIZ string specifying which user to remove from the group account.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.

Manifest	Value	Meaning
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_GroupNotFound	2220	The group does not exist.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_SpeGroupOp	2234	This operation is not allowed on this special group.
NERR_UserNotInGroup	2237	The user does not belong to this group.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize[-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite.

Remarks

Removing a user from a group does not delete the user's account in the system.

If an application tries to delete a user name from the special groups (USERS, ADMINS, or GUESTS), NetGroupDelUser returns the NERR_SpeGroupOp error code.

Related Information

For information on:

- Adding a user to a group—See "NetGroupAddUser" on page 3-115.
- Deleting a group—See "NetGroupDel" on page 3-118.
- Retrieving a list of members of a group—See "NetGroupGetUsers" on page 3-130.

NetGroupEnum

The NetGroupEnum (partially admin, DOS) function lists all group accounts on the user accounts subsystem (UAS) database.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGroupEnum(servername, level, buf, buflen,
             entriesread, totalentries)
char far *            servername;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  entriesread;
unsigned short far *  totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local server.
- *level* specifies the level of detail (0 or 1) for the *group_info* data structure.
- *buf* points to the *group_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries available.

On successful returns, *buf* contains *entriesread* number of *group_info* data structures.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_BAD_NETPATH	53	The network path cannot be found.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.

Manifest	Value	Meaning
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_GroupNotFound	2220	The group does not exist.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFNoRoom	2228	There are too many names in the access control file.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.

Manifest	Value	Meaning
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosRead
- DosSemClear.

Remarks

Only the group names can be retrieved with ordinary user's privilege. With administrative privilege, the comments can be returned.

Related Information

For information on:

- Adding a new group to the UAS database—See "NetGroupAdd" on page 3-112.
- Removing a group from the UAS database—See "NetGroupDel" on page 3-118.

NetGroupGetInfo

The NetGroupGetInfo (partially admin, DOS) retrieves group-related information.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGroupGetInfo (servername, groupname, level, buf, buflen,
                 totalavail)

char far *           servername;
char far *           groupname;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local server.
- *groupname* points to an ASCIIZ string specifying the group from which to get information.
- *level* specifies the level of detail (0 or 1) for the *group_info* data structure.
- *buf* points to the *group_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_GroupNotFound	2220	The group does not exist.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFNoRoom	2228	There are too many names in the access control file.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_SpeGroupOp	2234	This operation is not allowed on this special group.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

A user without administrative privilege can call this API only with level 0 on a remote call. Users cannot issue this function on the groups to which they do not belong.

NetGroupGetUsers

The NetGroupGetUsers (partially admin, DOS) function returns a list of members of a particular group in the user accounts subsystem (UAS) database. Users can perform this function on groups to which they belong.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGroupGetUsers(servername, groupname, level, buf, buflen,
                 entriesread, totalentries)
char far *           servername;
char far *           groupname;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far * entriesread;
unsigned short far * totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local server.
- *groupname* points to an ASCIIZ string specifying the name of the group whose members will be listed.
- *level* specifies the level of detail (0) for the *group_users_info_0* data structure.
- *buf* points to the *group_users_info_0* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_BAD_NETPATH	53	The network path cannot be found.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.

Manifest	Value	Meaning
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_GroupNotFound	2220	The group does not exist.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFNoRoom	2228	There are too many names in the access control file.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.

Manifest	Value	Meaning
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosRead
- DosSemClear.

Remarks

This is functionally equivalent to an Enum call because it enumerates the users in a group. See "NetGroupEnum" on page 3-124. NetGroupGetUsers of the special groups USERS, ADMINS, and GUESTS is an *admin* call.

Related Information

For information on:

- Listing all groups to which a user belongs—See "NetGroupGetUsers" on page 3-130.
- Listing the names of groups in the UAS database—See "NetGroupEnum" on page 3-124.

NetGroupSetInfo

The NetGroupSetInfo (admin, DOS) function sets group-related information.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGroupSetInfo (servername, groupname, level, buf, buflen,
                 parmnum)

char far *          servername;
char far *          groupname;
short               level;
char far *          buf;
unsigned short      buflen;
short               parmnum;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local server.
- *groupname* points to an ASCIIZ string specifying the group to set the information.
- *level* specifies the level of detail (1) for the *group_info* data structure.
- *buf* points to the data structure if *parmnum* is zero. Otherwise, *buf* points to the specific data component that will be changed.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* determines whether *buf* contains a complete *group_info* data structure or a single data structure component. If *parmnum* is 0, *buf* must contain the *group_info_1* data structure. The only settable field is *comment*.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_GroupNotFound	2220	The group does not exist.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_SpeGroupOp	2234	This operation is not allowed on this special group.

Manifest	Value	Meaning
NERR_UserInGroup	2236	The user already belongs to this group.
NERR_UserNotInGroup	2237	The user does not belong to this group.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

NetGroupSetUsers

The NetGroupSetUsers (admin, DOS) function sets information about users who belong to a group.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetGroupSetUsers (servername, groupname, level, buf, buflen, entries)

char far *          servername;
char far *          groupname;
short               level;
char far *          buf;
unsigned short      buflen;
unsigned short      entries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local server.
- *groupname* points to an ASCIIZ string specifying the group to set the users.
- *level* specifies the level of detail (0) for the *group_users_info* data structure.
- *buf* points to the *group_users_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entries* is the number of entries supplied in the buffer.

Buffer Contents on Call (format for a single entry):

Level 0 contains a *struct group_users_info_0*, repeated *entries* times.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_GroupNotFound	2220	The group does not exist.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFNoRoom	2228	There are too many names in the access control file.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.

Manifest	Value	Meaning
NERR_SpeGroupOp	2234	This operation is not allowed on this special group.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize[-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite.

Remarks

Trying to set the user membership of special group causes the NERR_SpeGroupOp error code to be returned.

Handle Category

NetHandleGetInfo (*local, server*)—See "NetHandleGetInfo" on page 3-140.

NetHandleSetInfo (*local, server*)—See "NetHandleSetInfo" on page 3-143.

Description

Two APIs are provided to get and set information on a per-handle basis. They are used with CHARDEV.H and NETCONS.H include files.

Remote Serial Device and Named Pipe Handles

The APIs provide per-handle control over the communications parameters for remote serial device and remote named pipe handles. These parameters are described in detail in "Requester Category" on page 3-208, under fields *wki0_chartime* and *wki0_charcount*.

The values in the *wksta_info_0* data structure are used as the default for each opened handle. The NetHandle APIs allow those parameters to be inspected and tuned on a per-handle basis.

Serving Side of Named Pipd Handles

The NetHandleGetInfo API is used to identify the user of a particular instance of a remote named pipe with multiple instances. If the named pipe has been opened locally, the error ERROR_INVALID_PARAMETER is returned.

Data Structures

```
struct handle_info_1 {
   unsigned long   hdli1_chartime;
   unsigned short  hdli1_charcount;
};
```

where:

- *hdli1_chartime* is the amount of time (in milliseconds) the requester collects data to send to a shared serial device queue or a named pipe.
- *hdli1_charcount* is the number of characters (in bytes) the requester stores before sending data to a serial device queue or a named pipe.

```
struct handle_info_2 {
   char far *   hdli2_username;
};
```

where:

- *hdli2_username* is the user name of the user attached to a named pipe. It can be applied to a handle of the serving side of a valid remote named pipe only.

Related Information

For information on:

- Creating multiple queues for a particular serial device—See "Serial Device Category" on page 3-238.
- Data structure architecture—See Chapter 1, "Overview of OS/2 LAN Server API."
- Include files—See Appendix A, "Include Files."

NetHandleGetInfo

The NetHandleGetInfo (local, server) function retrieves handle-specific information.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetHandleGetInfo(handle, level, buf, buflen, totalavail)
unsigned short        handle;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  totalavail;
```

where:

- *handle* is a unique identification of a communication device queue or a named pipe.
- *level* specifies the level of detail (1 or 2) to be returned in the *handle_info* data structure.
- *buf* points to the *handle_info_1* or *handle_info_2* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to the unsigned short integer indicating the number of bytes of information available.

Serial Device and Named Pipe Handles Information (Level 1)

Level 1 information is available for handles to remote serial devices and remote named pipes. If *level* is 1 and the return code is 0 (NERR_Success), the buffer contains a *handle_info_1* data structure.

Named Pipe Handles Information (Level 2)

Level 2 information is available for named pipe handles. If *level* is 2 and the return code is 0 (NERR_Success), the buffer contains a *handle_info_2* data structure.

Under DOS, only level one is valid, and only on a handle to a remote named pipe.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SHARING_VIOLATION	32	A sharing violation occurred.

Manifest	Value	Meaning
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosDevIOCtl
- DosFsRamSemClear
- DosRamSemRequest
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)

- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosQNmPipeInfo
- DosSemClear
- DosSemRequest.

Remarks

When level 1 is specified, the function has to be run on the requester. If the handle is not to a remote serial device or remote named pipe, the error ERROR_INVALID_PARAMETER is returned.

When level 2 is specified, this function has to be run on the server. On level 2, it is used to identify the user of a particular instance of a remote named pipe with multiple instances. If the handle is not to a named pipe that a remote client currently has open, the error ERROR_INVALID_PARAMETER is returned.

NetHandleSetInfo

The NetHandleSetInfo (local, server) function sets handle-specific information.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetHandleSetInfo(handle, level, buf, buflen, parmnum)
unsigned short        handle;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short        parmnum;
```

where:

- *handle* is a unique identification of a communication device queue or a named pipe.
- *level* specifies the level of detail (1) to be returned in the *handle_info* data structure.
- *buf* points to the *handle_info_1*data structure or a single data structure component.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* determines whether *buf* contains a complete *handle_info* data structure or a single component. If *parmnum* is 0 and *level* is 1 then *buf* must contain a complete *handle_info_1* or *handle_info_2* data structure. Otherwise, *parmnum* must specify the ordinal position value for one of the following data structure components, as defined in CHARDEV.H as follows:

Manifest	Value	Component
HANDLE_SET_CHAR_TIME	1	*hdli1_chartime*
HANDLE_SET_CHAR_COUNT	2	*hdli1_charcount*

Serial Device and Named Pipe Handles Information (Level 1)

For this function, only level 1 information is valid. Level 1 is valid for handles to remote serial devices and remote named pipes.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SHARING_VIOLATION	32	A sharing violation occurred.

Manifest	Value	Meaning
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosDevIOCtl
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

Because only level 1 is allowed in calling this function, it has to be run on a requester. If the handle is not to a remote serial device or remote named pipe, the error ERROR_INVALID_PARAMETER is returned.

Under DOS, only level 1 is valid, and only on a handle to a remote named pipe.

Mailslot Category

DosDeleteMailslot (*local, DOS*)—See "DosDeleteMailslot" on page 3-148.

DosMailslotInfo (*local, DOS*)—See "DosMailslotInfo" on page 3-149.

DosMakeMailslot (*local, DOS*)—See "DosMakeMailslot" on page 3-150.

DosPeekMailslot (*local, DOS*)—See "DosPeekMailslot" on page 3-151.

DosReadMailslot (*local, DOS*)—See "DosReadMailslot" on page 3-152.

DosWriteMailslot (*local, DOS*)—See "DosWriteMailslot" on page 3-154.

The functions in the Mailslot category provide one-way interprocess communication (IPC). They are used with the MAILSLOT.H and NETCONS.H include files.

Description

Through OS/2 LAN Requester/Server mailslots, data can be sent to either local or remote applications on the network. The Mailslot functions create and delete mailslots, retrieve information about a mailslot or a message in it, and write messages to mailslots.

An application creates a mailslot on a local computer by calling the DosMakeMailslot function and assigning the mailslot a name in the format:

\mailslot*name*

where:

- *name* is a unique set of characters distinguishing the mailslot from other mailslots on the computer.

The DosMakeMailslot function returns a handle to the mailslot. This handle can then be used with DosPeekMailslot to read a message in a mailslot, with DosReadMailslot to read and remove a message, with DosMailslotInfo to return information on a mailslot, and with DosDeleteMailslot to delete a mailslot.

Any application can write messages to any mailslot on any computer on the network by calling the DosWriteMailslot function. DosWriteMailslot accepts mailslot names both in a local and remote format, as follows:

Format	Type
\mailslot*name*	Local mailslot
*computername***mailslot***name*	Remote mailslot

To write data to a mailslot on a remote computer, the name of the mailslot must also include a computer name. This requirement enables multiple remote computers to use the same mailslot name locally, but to have different names on the network (the computer name must be unique).

An application can write the same message to all computers on the network that have a mailslot of a particular name. Only the second-class delivery is provided. By specifying an asterisk (*) for the computer name when calling NetWriteMailslot,

***\mailslot\name**

sends the same message to the named mailslot on every computer in the sender's primary domain that has the locally created mailslot. There is one limitation: requesters can only receive second-class messages of up to 400 bytes in length. Servers can receive first-class or second-class messages of any size.

Two classes of messages—*first-class* and *second-class*—can be sent to mailslots.

First-class messages, limited to mailslots on local computers and remote servers, are guaranteed — the message will be delivered or the sender will be notified. If a mailslot is full when a first-class message arrives, DosWriteMailslot waits until DosReadMailslot reads and removes a message from the mailslot or until the delivery time out expires (controlled by the *timeout* parameter in the DosWriteMailslot function).

Second-class messages are simply sent; no return code informs the sender of an unsuccessful delivery. This simpler delivery system tends to make second-class messages faster than first-class messages.

Messages are stored in the mailslot according to when they were received and the *priority* assigned them. Each message is assigned a priority from 0 (low) through 9 (high) by way of the *priority* parameter of the DosWriteMailslot function. Generally, these priorities dictate the order in which messages are stored in a mailslot. High-priority messages are placed ahead of previously stored messages with the same or lower priority. However, since the OS/2 program is a multi-tasking operating system, this scheme cannot be guaranteed at any one time.

Mailslot messages can be read only by the process that created the mailslot.

The DosReadMailslot function reads and then removes the most current (next available) message. Since new messages may be placed in front of other messages due to priority, a process cannot be guaranteed that a message read by DosReadMailslot will be the same message seen earlier by DosPeekMailslot.

DOS Considerations

Under DOS, the functions can be executed on a local requester. Note that mailslots can only be read or deleted by the process that created them. Mailslots created by a process are deleted when that process ends.

Related Information

For information on interprocess communications (IPC), see "Named Pipe Category" on page 3-191.

DosDeleteMailslot

The DosDeleteMailslot (local, DOS) function deletes a mailslot, discarding all messages, whether or not they have been read.

Syntax

```
#include <netcons.h>
#include <mailslot.h>

unsigned far pascal
DosDeleteMailslot(handle)
unsigned handle;
```

where:

- *handle* specifies the mailslot (by its handle) to delete.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_INVALID_HANDLE	6	The specified handle is not valid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Remarks

Mailslots enable applications to create and store messages during execution. Generally, these mailslots are deleted as the last step in the execution of a program.

A mailslot can be deleted only by the application that created it.

Related Information

For information on:

- Creating a mailslot—See "DosMakeMailslot" on page 3-150.
- Obtaining information on the status of a mailslot—See "DosMailslotInfo" on page 3-149.

DosMailslotInfo

The DosMailslotInfo (local, DOS) function retrieves information about a particular mailslot.

Syntax

```
#include <netcons.h>
#include <mailslot.h>

unsigned far pascal
DosMailslotInfo(handle, messagesize, mailslotsize,
                nextsize, nextpriority, msgcount)
unsigned            handle;
unsigned short far * messagesize;
unsigned short far * mailslotsize;
unsigned short far * nextsize;
unsigned short far * nextpriority;
unsigned short far * msgcount;
```

where:

- *handle* specifies which mailslot (by its handle) to return information about.
- *messagesize* points to an unsigned short integer indicating the maximum size (in bytes) of message that the mailslot can accept.
- *mailslotsize* points to an unsigned short integer indicating the size (in bytes) of the mailslot. *mailslotsize* must equal or exceed *messagesize*.
- *nextsize* points to an unsigned short integer indicating the size (in bytes) of the next message in the mailslot. If 0, no message is available.
- *nextpriority* points to an unsigned short integer indicating the priority (0 theough 9) of the next message in the mailslot (undefined if *nextsize* is 0).
- *msgcount* points to an unsigned short integer indicating the number of messages the mailslot contains.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_INVALID_HANDLE	6	The specified handle is not valid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Related Information

For information on:

- Creating (and obtaining the handle for) a mailslot—See "DosMakeMailslot" on page 3-150.
- Writing a message to a mailslot—See "DosWriteMailslot" on page 3-154.
- Retrieving the most current message in a mailslot—See "DosReadMailslot" on page 3-152.

DosMakeMailslot

The DosMakeMailslot (local, DOS) function creates a mailslot and returns its handle.

Syntax

```
#include <netcons.h>
#include <mailslot.h>

unsigned far pascal
DosMakeMailslot(name, messagesize, mailslotsize, handle)
char far *     name;
unsigned short messagesize;
unsigned short mailslotsize;
unsigned far * handle;
```

where:

- *name* points to an ASCIIZ string assigning a name to the mailslot. Use the format **\mailslot***name*.
- *messagesize* specifies the maximum message size (in bytes) that the mailslot can accept. Generally, mailslots cannot accept messages larger than 65475 bytes.
- *mailslotsize* specifies the size (in bytes) of the mailslot. *mailslotsize* must equal or exceed *messagesize*.
- *handle* points to an unsigned integer that is the returned handle for the mailslot.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_PATH_NOT_FOUND	3	The path was not found.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocShrSeg
- DosExitList.

Remarks

Mailslot names must be unique; no two mailslots on any one computer can have the same name.

Mailslot handles cannot be passed to other processes by way of the OS/2 DosExecPgm function; however, mailslot handles can be shared among threads in a single process. Thus, multiple threads can use the same handle to read or write data to the mailslot.

DosPeekMailslot

The DosPeekMailslot (local, DOS) function reads the next available message in a mailslot without removing it.

Syntax

```
#include <netcons.h>
#include <mailslot.h>

unsigned far pascal
DosPeekMailslot(handle, buf, bytesread, nextsize, nextpriority)
unsigned            handle;
char far *          buf;
unsigned short far * bytesread;
unsigned short far * nextsize;
unsigned short far * nextpriority;
```

where:

- *handle* specifies which mailslot (by its handle) is to be read.
- *buf* points to the returned message. *buf* must be as large as the *messagesize* parameter passed to the DosMakeMailslot function.
- *bytesread* points to an unsigned short integer indicating the size (in bytes) of the returned message. If no message is available, *bytesread* is 0.
- *nextsize* points to an unsigned short integer indicating the size (in bytes) of the next message in the mailslot. If the mailslot contains no other message, *nextsize* is 0.
- *nextpriority* points to an unsigned short integer indicating the priority of the next message in the mailslot (undefined if *nextsize* is 0).

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_INVALID_HANDLE	6	The specified handle is not valid.
ERROR_BROKEN_PIPE	109	Write on pipe with no reader.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Other error return codes may be returned from the DosSemRequest function.

Remarks

If a higher-priority message arrives, there is no guarantee that a message previously read by the DosPeekMailslot function will be the same message read by a subsequent call to the DosReadMailslot function.

Related Information

For information on:

- Reading and removing a message—See "DosReadMailslot" on page 3-152.
- Writing a message to a mailslot—See "DosWriteMailslot" on page 3-154.

DosReadMailslot

The DosReadMailslot (local, DOS) function reads, then removes the next available message of a mailslot.

Syntax

```
#include <netcons.h>
#include <mailslot.h>

unsigned far pascal
DosReadMailslot(handle, buf, bytesread, nextsize,
                nextpriority, timeout)
unsigned              handle;
char far *            buf;
unsigned short far *  bytesread;
unsigned short far *  nextsize;
unsigned short far *  nextpriority;
long                  timeout;
```

where:

- *handle* specifies which mailslot (by its handle) to read from.
- *buf* points to the returned message. *buf* must be as large as the *messagesize* parameter passed to the DosMakeMailslot function.
- *bytesread* points to an unsigned short integer indicating the size (in bytes) of the returned message. If 0, no message is available.
- *nextsize* points to an unsigned short integer indicating the size (in bytes) of the next message in the mailslot. If 0, the mailslot contains no more messages.
- *nextpriority* points to an unsigned short integer indicating the priority (0-9) of the next message (undefined if *nextsize* is 0.)
- *timeout* points to an unsigned short integer indicating the number of milliseconds to wait if a message is not available immediately. If 0, DosReadMailslot does not wait; if -1, DosReadMailslot waits indefinitely.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_INVALID_HANDLE	6	The specified handle is not valid.
ERROR_INTERRUPT	95	A system call has been interrupted.
ERROR_BROKEN_PIPE	109	Write on pipe with no reader.
ERROR_SEM_TIMEOUT	121	A time out happened from the semaphore API functions.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Other error return codes may be returned from the DosSemRequest function.

Remarks

Messages are stored in a mailslot based on their priority (0-9). An incoming message with a higher priority may be stored ahead of a previously stored message with the same or lower priority. The message read and removed by DosReadMailslot is always the next available.

Related Information

For information on:

- Reading a message without removing it—See "DosPeekMailslot" on page 3-151.
- Writing a message to a mailslot—See "DosWriteMailslot" on page 3-154.

DosWriteMailslot

The DosWriteMailslot (local, DOS) function writes a message to a particular mailslot.

Syntax

```
#include <netcons.h>
#include <mailslot.h>

unsigned far pascal
DosWriteMailslot(name, message, size, priority, class, timeout)
char far *     name;
char far *     message;
unsigned short size;
unsigned short priority;
unsigned short class;
long           timeout;
```

where:

- *name* points to an ASCIIZ string containing the name of the mailslot to which the message is to be written. For a local mailslot, use the format **mailslot***name*. Use the *computername***mailslot***name* format for a remote mailslot. Use ***mailslot***name* for all mailslots with the same name, but on different computers in the primary domain.
- *message* points to an ASCIIZ string containing the message to be written to the mailslot.
- *size* specifies the size (in bytes) of *message*.
- *priority* assigns a priority (0 through 9) to the message. High-priority messages are generally placed ahead of previously stored messages with lower priority.
- *class* specifies the class of mail service to be provided.
  - First-class mail (*class* is 1) forces DosWriteMailslot to wait until a mailslot has enough room to accept *message* or until *timeout* expires. First-class mail can be delivered only to remote servers or local computers.
  - Second-class mail (*class* is 2) causes DosWriteMailslot to fail if there is not enough room to write *message* in the mailslot. Second-class mail can be delivered to requesters and servers.
- *timeout* specifies the number of milliseconds to attempt writing a message to a mailslot. If 0, DosWriteMailslot attempts to write the message only once. If -1, DosWriteMailslot attempts to write a message to a mailslot for an indefinite time.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_PATH_NOT_FOUND	3	The path was not found.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.

Manifest	Value	Meaning
ERROR_BAD_FORMAT	11	The format is not valid.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_NETWORK_BUSY	54	The network is busy.
ERROR_BAD_NET_NAME	67	This network name cannot be found.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INTERRUPT	95	A system call has been interrupted.
ERROR_BROKEN_PIPE	109	Write on pipe with no reader.
ERROR_BUFFER_OVERFLOW	111	The buffer passed to system call is too small to hold return data.
ERROR_SEM_TIMEOUT	121	A time out happened from the semaphore API functions.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosDevIOCtl
- DosFSCtl
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- DosSemRequest
- redir.GetBiosInfo[-ERROR_NO_MORE_FILES]
- redir.NETTRANSACTION_1
- redir.NIOCBIOSOPEN.

Remarks

To send a message to all computers on the primary domain that have a local mailslot with the same name, an application must pass the *name* parameter ***mailslot**name* and the *class* parameter 2 to DosWriteMailslot.

Second-class messages must be 400 bytes or smaller when written to remote requesters; they can be any size when written to local computers or remote servers.

Related Information

For information on:

- Creating a mailslot—See "DosMakeMailslot" on page 3-150.
- Reading a message—See "DosReadMailslot" on page 3-152.

Message Category

NetMessageBufferSend (*admin, DOS*)—See "NetMessageBufferSend" on page 3-161.

NetMessageFileSend (*admin, DOS*)—See "NetMessageFileSend" on page 3-164.

NetMessageLogFileGet (*admin, DOS*)—See "NetMessageLogFileGet" on page 3-168.

NetMessageLogFileSet (*admin, DOS*)—See "NetMessageLogFileSet" on page 3-170.

NetMessageNameAdd (*admin, DOS*)—See "NetMessageNameAdd" on page 3-173.

NetMessageNameDel (*admin, DOS*)—See "NetMessageNameDel" on page 3-176.

NetMessageNameEnum (*admin, DOS*)—See "NetMessageNameEnum" on page 3-179.

NetMessageNameFwd (*admin*)—See "NetMessageNameFwd" on page 3-182.

NetMessageNameGetInfo (*admin, DOS*)—See "NetMessageNameGetInfo" on page 3-185.

NetMessageNameUnFwd (*admin*)—See "NetMessageNameUnFwd" on page 3-188.

The functions in the Message category are used to send, log, and forward messages. The administrator can execute these functions remotely. They are used with the MESSAGE.H and NETCONS.H include files.

Description

A *message* is any file or buffer of data sent to a messaging name on the network. To receive a message, a user or application must register a messaging name (using the NetMessageNameAdd function) in the message name table of a computer. A message name table contains a list of registered messaging names permitted to receive messages and a list of users and applications to which a message can be forwarded. Messaging names must be case-sensitive and unique on the physical network, not just the domain.

Messaging names are deleted from the message name table using the NetMessageNameDel function.

To list all the names stored in the table, an application can call the NetMessageNameEnum function. For information on a particular user in the table, an application can call the NetMessageNameGetInfo function.

All of the Message functions except NetMessageBufferSend and NetMessageFileSend require that the local computer be running the *messenger* service. The NetMessageBufferSend and the NetMessageFileSend functions only require that the *remote* computer receiving a message be running the *messenger* service.

To send a message to a user, an application can call either the NetMessageFileSend function (to send a file) or the NetBufferSend function (to send a buffer of information).

All messages sent to a user on a particular computer can be forwarded to another user on a different computer using the NetMessageNameFwd function. The NetMessageNameUnFwd function is used to end message forwarding.

Applications can also send broadcast messages to all users on the network registered in the message name table of each computer by passing the *name* parameter for the NetMessageFileSend function or NetMessageBufferSend function as an asterisk (*).

To send to all users on a particular domain, pass the *name* parameter as "domain*."

Users can receive messages in one of two ways (or both at the same time):

- The received message is logged to a *message log file* and looked at later.
- The message is displayed as a popup message on the screen. To receive a popup, the *netpopup* service must be started. For more information on starting the *netpopup* service, see "Service Category" on page 3-298.

If an application turns logging on (using NetMessageLogFileSet), all messages received for a particular user are stored in a message log file. The NetMessageLogFileGet function returns the name of a message log file of a requester or server and indicates whether or not message logging is enabled. The default message log file is \IBMLAN\LOGS\MESSAGES.LOG.

The message log file contains a message in the following format:

- A header specifying who sent the message, who received the message, and when (time and date) the message was received
- A blank line
- The contents of the message
- A blank line
- A line containing four asterisks (*)
- A blank line.

For example, the following is the contents of the message log file containing two messages:

```
Message from KRISCA to AJSCHEL on Aug 04, 1990, 14:05:20

Hello, this is a BUFFER message.

****

Message from KRISCA to AJSCHEL on Aug 04, 1990, 14:11:48

Hello, this is a FILE message.
```

Note: Any process opening the message log file must open it in only the read-only deny-none mode; otherwise, the *messenger* service fails when trying to log incoming messages.

DOS Considerations

Under DOS, the functions can be executed only on a local requester. Attempting to execute the functions on a remote server returns ERROR_NOT_SUPPORTED.

Messages cannot be forwarded, unforwarded, or logged under DOS.

By default, the DOS LAN Requester accepts only two names in the message name table—the name of the requester and of the user. To define more names, edit the DOSLAN.INI file and change the value of the *nmsg* parameter for the messenger component. For more information on the DOSLAN.INI file, see the *DOS LAN Requester User's Guide*.

The maximum size of a message under DOS is 64KB.

Data Structures

The NetMessageNameEnum and NetMessageNameGetInfo functions can accept or return data at a level 0 or level 1 of detail using the following data structures. None of the other Message functions use a data structure.

Message Information (Level 0)

```
struct msg_info_0 {
    char msgi0_name[CNLEN+1];
};
```

where:

- *msgi0_name* is an ASCIIZ string specifying which messaging name to send the message.

Message Information (Level 1)

```
struct msg_info_1 {
   char          msgi1_name[CNLEN+1];
   char          msgi1_forward_flag;
   unsigned char msgi1_pad1;
   char          msgi1_forward[CNLEN+1];
};
```

where:

- *msgi1_name* is an ASCIIZ string specifying which messaging name to send the message.
- *msgi1_forward_flag* specifies whether messages will be sent to a user or application on the local computer, or forwarded to a user or application on a remote computer. *msgi1_forward_flag* can be defined as follows:

Bit	Manifest	Meaning
0-1	MSGNAME_NOT_FORWARDED	Reserved; must be 0.
2	MSGNAME_FORWARDED_TO	If 1, specifies a user name on a remote computer.
3	MSGNAME_NOT_FORWARDED	Reserved; must be 0.

Bit	Manifest	Meaning
4	MSGNAME_FORWARDED_FROM	If 1, specifies a user name on the local computer.
5-7	MSGNAME_NOT_FORWARDED	Reserved; must be 0.

- *msgil_padl* WORD-aligns the data structure components.
- *msgil_forward* is an ASCIIZ string specifying the user name to which the message will be sent, if messages are to be forwarded.

Related Information

For information on starting services and the *messenger* service see "Service Category" on page 3-298.

NetMessageBufferSend

The NetMessageBufferSend (admin, DOS) function sends a buffer of information to a registered messaging name.

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageBufferSend (servername, name, buf, buflen)
const char far * servername;
char far *       name;
char far *       buf
unsigned short   buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *name* points to an ASCIIZ string indicating the name of the registered user or application to receive the message. To broadcast a message to all requesters on the LAN, have *name* point to an asterisk (*). To broadcast a message to a domain, have *name* point to a domain name, followed by an asterisk.
- *buf* points to the message.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_REM_NOT_LIST	51	This remote computer is not listening.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.

Manifest	Value	Meaning
NERR_NoNetworkResource	2105	A network resource shortage occurred.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_NetworkError	2136	A general network error has occurred.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NoComputerName	2270	A computer name has not been configured.
NERR_NameNotFound	2273	The message alias cannot be found on the local area network.
NERR_AlreadyExists	2276	This message alias already exists locally.
NERR_TooManyNames	2277	The maximum number of added message aliases has been exceeded.
NERR_PausedRemote	2281	The message has been sent but the reception is currently paused.
NERR_BadReceive	2282	The message was sent but not received.
NERR_NotLocalName	2285	The name is not on the local computer.
NERR_TruncatedBroadcast	2289	The broadcast message was truncated.
NERR_DuplicateName	2297	A duplicate message alias exists on the local area network.

Manifest	Value	Meaning
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- DosMailslot.

Remarks

For broadcast messages (*name* points to "*" or "domain (*)"), the message can be no longer than 128 characters (and is not guaranteed to be delivered). For messages sent to all computers in a domain, the limit is 128 bytes. Otherwise, the message can be any length, provided it does not exceed the maximum receivable message size for that computer, which is set with the *sizmessbuf* parameter in IBMLAN.INI. (The *sizmessbuf* parameter in IBMLAN.INI cannot define a value larger than 64KB.) The default value of *sizmessbuf* on any server is 4KB. Note that the total size of *sizmessbuf* may be divided between different messages in its heap (if messages are arriving at the same time), reducing the actual size of any one message that can be received. And, note that the *sizmessbuf* parameter can accept only limited values. For more information on the IBMLAN.INI file, see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*

NetMessageBufferSend does not require the *messenger* service to be started on a local computer. The remote computer needs the *messenger* service to be started.

DOS Considerations

Under DOS, the name parameter cannot point to the name of the local requester or to the user currently logged onto that requester.

Related Information

For information on:

- Adding a user to a message table—See "NetMessageNameAdd" on page 3-173.
- The *messenger* service—See "Service Category" on page 3-298.
- Sending a message file to a user—See "NetMessageFileSend" on page 3-164.
- Setting the *sizmessbuf* parameter of a server in IBMLAN.INI—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*

NetMessageFileSend

The NetMessageFileSend (admin, DOS) function sends a file to a registered messaging name.

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageFileSend (servername, name, filespec)
const char far * servername;
char far *       name;
char far *       filespec;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies the local computer.
- *name* points to an ASCIIZ string specifying the registered user or application to receive the file. To broadcast a file to all registered users and applications, pass the *name* parameter as a pointer to the ASCII string "*." To broadcast a message to a domain, have *name* point to a domain name, followed by an asterisk (*).
- *filespec* points to an ASCIIZ string [d:][\ path\]file[.ext] specifying the path name of a file to send.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_REM_NOT_LIST	51	This remote computer is not listening.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.

Manifest	Value	Meaning
NERR_NoNetworkResource	2105	A network resource shortage occurred.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_NetworkError	2136	A general network error has occurred.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NoComputerName	2270	A computer name has not been configured.
NERR_NameNotFound	2273	The message alias cannot be found on the local area network.
NERR_AlreadyExists	2276	This message alias already exists locally.
NERR_TooManyNames	2277	The maximum number of added message aliases has been exceeded.
NERR_PausedRemote	2281	The message has been sent but the reception is currently paused.
NERR_BadReceive	2282	The message was sent but not received.
NERR_NotLocalName	2285	The name is not on the local computer.
NERR_TruncatedBroadcast	2289	The broadcast message was truncated.
NERR_FileError	2290	An error occurred in reading the message file.

Manifest	Value	Meaning
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosOpen
- DosRead
- DosSemClear
- DosWriteMailslot.

Remarks

For broadcast messages to the physical network (*name* points to "*" or "domain (*)"), the message can be no longer than 128 characters (and is not guaranteed to be delivered). For messages sent to all computers in a domain, the limit is 128 bytes. Otherwise, the message can be any length, provided it does not exceed the maximum receivable message size for that computer, which is set with the *sizmessbuf* parameter in IBMLAN.INI. (The IBMLAN.INI *sizmessbuf* parameter cannot define a value larger than 64KB.) The default value of *sizmessbuf* on any server is 4KB. Note that the total size of *sizmessbuf* may be divided between different messages in its heap (if messages are arriving at the same time), reducing the actual size of any one message that can be received. And, note that the *sizmessbuf* parameter can also accept only limited values. For more information on the IBMLAN.INI file, see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide*.

NetMessageFileSend does not require the *messenger* service to be started on a local computer.

If any special characters (for example, Ctrl + Z) are sent in a file, no information is omitted.

DOS Considerations

Under DOS, the name parameter cannot point to the name of the local requester or to the user currently logged on to that requester.

Related Information

For information on:

- The *messenger* service—See "Service Category" on page 3-298.
- Sending a buffer of information to a user—See "NetMessageBufferSend" on page 3-161.
- Setting the *sizmessbuf* parameter of a server in IBMLAN.INI—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*

NetMessageLogFileGet

The NetMessageLogFileGet (admin, DOS) function retrieves the name of the message log file and the current logging status (on or off).

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageLogFileGet (servername, buf, buflen, on)
const char far * servername;
char far *       buf
unsigned short   buflen;
short far *      on;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *buf* points to the returned message log file path name.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *on* points to a short integer specifying whether or not logging is enabled. If zero, message logging is disabled. If non-zero, message logging is enabled.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.

Manifest	Value	Meaning
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_MsgNotStarted	2284	The messenger service has not been started.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosSemClear.

Remarks

NetMessageLogFileGet requires that the *messenger* service be started.

Related Information

For information on:

- The *messenger* service—See "Service Category" on page 3-298.
- Modifying the name and the logging status of the message log file—See "NetMessageLogFileSet" on page 3-170.

NetMessageLogFileSet

The NetMessageLogFileSet (admin, DOS) function specifies a file to log messages received by registered users and enables or disables logging.

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageLogFileSet (servername, filespec, on)
const char far * servername;
char far *       filespec;
unsigned short   on;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *filespec* points to an ASCIIZ string specifying the path name of the device (LPTn or COMn) or file to which the messages are logged.

  If *filespec* is passed as a NULL pointer, the name of the current message log file does not change. If *filespec* points to a NULL string (""), no message file will be used; in this case, the value of *on* must be 0.

  If *filespec* points to a relative path, the path must be relative to the IBMLAN\LOGS directory. All other path names must be fully qualified. If no file name extension is provided, the .LOG file extension is appended.
- *on* is a short integer specifying whether or not logging is enabled. If zero, message logging is disabled. If non-zero, message logging is enabled.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
ERROR_FILENAME_EXCED_RANGE	206	The file name is longer than 8 characters or the extension is longer than 3 characters.
ERROR_VIO_DETACHED	465	The console is not available for logging.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RedirectedPath	2117	The operation is invalid on a redirected device.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_MsgNotStarted	2284	The messenger service has not been started.
NERR_InvalidDevice	2294	This is an invalid device.
NERR_WriteFault	2295	A write fault has occurred.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CantType	2357	The type of input cannot be determined.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosOpen
- DosQHandType
- DosSemClear
- DosWrite
- redir.GetNetInitPath.

Remarks

NetMessageLogFileSet requires that the *messenger* service be started.

Related Information

For information on:

- The *messenger* service—See "Service Category" on page 3-298.
- Retrieving the name and logging status of the message log file—See "NetMessageLogFileGet" on page 3-168.

NetMessageNameAdd

The NetMessageNameAdd (admin, DOS) function registers a name in the message name table.

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageNameAdd(servername, name, fwd_action)
const char far * servername;
char far *       name;
short            fwd_action;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *name* points to an ASCIIZ string specifying a name to add to the message name table.
- *fwd_action* specifies the action to take if *name* is already forwarded. If *fwd_action* is non-zero, the name is added to the message name table; a zero value causes an error to be returned if the name has already been forwarded.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_REM_NOT_LIST	51	This remote computer is not listening.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_NoNetworkResource	2105	A network resource shortage occurred.

Manifest	Value	Meaning
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_NetworkError	2136	A general network error has occurred.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NameNotFound	2273	The message alias cannot be found on the local area network.
NERR_AlreadyForwarded	2274	This message alias has already been forwarded.
NERR_AddForwarded	2275	This message alias has been added but is still forwarded.
NERR_AlreadyExists	2276	This message alias already exists locally.
NERR_TooManyNames	2277	The maximum number of added message aliases has been exceeded.
NERR_MsgNotStarted	2284	The messenger service has not been started.
NERR_DuplicateName	2297	A duplicate message alias exists on the local area network.
NERR_DeleteLater	2298	This message alias will be deleted later.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFSRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Related Information

For information on:

- Deleting a user name from a message name table—See "NetMessageNameDel" on page 3-176.
- Forwarding messages—See "NetMessageNameFwd" on page 3-182.
- Listing the user names in a message name table—See "NetMessageNameEnum" on page 3-179.
- *Messenger* service—See "Service Category" on page 3-298.

NetMessageNameDel

The NetMessageNameDel (admin, DOS) function deletes a name from a message name table.

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageNameDel(servername, name, fwd_action)
const char far * servername;
char far *       name;
short            fwd_action;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *name* points to an ASCIIZ string specifying the name to be removed.
- *fwd_action* specifies the action to take if the messages for *name* are being forwarded to another name. If *fwd_action* is non-zero, the forwarded name is deleted. A zero value prevents the name from being deleted.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_REM_NOT_LIST	51	This remote computer is not listening.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_NoNetworkResource	2105	A network resource shortage occurred.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.

Manifest	Value	Meaning
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_NetworkError	2136	A general network error has occurred.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NameNotFound	2273	The message alias cannot be found on the local area network.
NERR_AlreadyForwarded	2274	This message alias has already been forwarded.
NERR_AlreadyExists	2276	This message alias already exists locally.
NERR_TooManyNames	2277	The maximum number of added message aliases has been exceeded.
NERR_DelComputerName	2278	The computer name cannot be deleted.
NERR_NameInUse	2283	The message alias is currently in use—try again later.
NERR_MsgNotStarted	2284	The messenger service has not been started.
NERR_NotLocalName	2285	The name is not on the local computer.
NERR_DuplicateName	2297	A duplicate message alias exists on the local area network.
NERR_DeleteLater	2298	This message alias will be deleted later.

Manifest	Value	Meaning
NERR_IncompleteDel	2299	The message alias was not successfully deleted from all networks.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

NetMessageNameDel requires that the *messenger* service be started.

Related Information

For information on:

- Adding a name to a message name table—See "NetMessageNameAdd" on page 3-173.
- Listing the names in a particular message name table—See "NetMessageNameEnum" on page 3-179.
- The *messenger* service—See "Service Category" on page 3-298.

NetMessageNameEnum

The NetMessageNameEnum (admin, DOS) function lists the name entries in a message name table.

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageNameEnum(servername, level, buf, buflen,
                   entriesread, totalentries)
const char far *     servername;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far * entriesread;
unsigned short far * totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail (0 or 1) to be returned in the *msg_info* data structure.
- *buf* points to the *msg_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_MsgNotStarted	2284	The messenger service has not been started.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

NetMessageNameEnum requires that the *messenger* service be started.

Related Information

For information on:

- Adding a name to a message name table—See "NetMessageNameAdd" on page 3-173.
- Deleting a name from a message name table—See "NetMessageNameDel" on page 3-176.
- The *messenger* service—See "Service Category" on page 3-298.
- Retrieving information about a user's message account—See "NetMessageNameGetInfo" on page 3-185.

NetMessageNameFwd

The NetMessageNameFwd (admin) function modifies the message name table to forward messages to another messaging name.

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageNameFwd(servername, name, forwardname, delfor)
const char far * servername;
const char far * name;
char far *       forwardname;
short            delfor;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *name* points to an ASCIIZ string specifying the name receiving messages.
- *forwardname* points to an ASCIIZ string specifying the name to receive *name's* forwarded messages.
- *delfor* specifies the action to take if *name* forwards messages to another name. If non-zero, then any previous forwarded user name is deleted; if 0, it is not deleted and an error is returned.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_REM_NOT_LIST	51	This remote computer is not listening.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_NoNetworkResource	2105	A network resource shortage occurred.

Manifest	Value	Meaning
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_NetworkError	2136	A general network error has occurred.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NameNotFound	2273	The message alias cannot be found on the local area network.
NERR_AlreadyForwarded	2274	This message alias has already been forwarded.
NERR_AlreadyExists	2276	This message alias already exists locally.
NERR_TooManyNames	2277	The maximum number of added message aliases has been exceeded.
NERR_LocalForward	2279	Messages cannot be forwarded back to the same workstation.
NERR_NameInUse	2283	The message alias is currently in use—try again later.
NERR_MsgNotStarted	2284	The messenger service has not been started.
NERR_NotLocalName	2285	The name is not on the local computer.
NERR_RemoteFull	2287	The message alias table on the remote station is full.

Manifest	Value	Meaning
NERR_DuplicateName	2297	A duplicate message alias exists on the local area network.
NERR_DeleteLater	2298	This message alias will be deleted later.
NERR_MultipleNets	2300	This operation is not supported on machines with multiple networks.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

NetMessageNameFwd requires that the *messenger* service be started.

Related Information

For information on:

- Listing the entries in the message name table of a server—See "NetMessageNameEnum" on page 3-179.
- The *messenger* service—See "Service Category" on page 3-298.
- Setting the *sizmessbuf* parameter for a server in IBMLAN.INI—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- Stopping forwarding of a user's messages—See "NetMessageNameUnFwd" on page 3-188.

NetMessageNameGetInfo

The NetMessageNameGetInfo (admin, DOS) function retrieves information about a user's entry in the message name table.

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageNameGetInfo(servername, name, level, buf,
                      buflen, totalavail)
const char far *      servername;
const char far *      name;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *name* points to an ASCIIZ string specifying the name of the user of interest.
- *level* specifies the level of detail (0 or 1) requested for the returned *msg_info* data structure.
- *buf* points to the *msg_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_MsgNotStarted	2284	The messenger service has not been started.
NERR_NotLocalName	2285	The name is not on the local computer.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

Data returned at a level of 0 provides only the name of the user. A level 1 structure provides the name of the user and whether or not message forwarding is available, and if so, to whom the messages are forwarded.

NetMessageNameGetInfo requires that the *messenger* service be started.

Related Information

For information on listing all user name entries in a message name table, see "NetMessageNameEnum" on page 3-179.

NetMessageNameUnFwd

The NetMessageNameUnFwd (admin) function stops forwarding a user's messages to another user.

Syntax

```
#include <netcons.h>
#include <message.h>

unsigned far pascal
NetMessageNameUnFwd(servername, name)
const char far * servername;
const char far * name;
```

where:

- *servername* points to an ASCIIZ string containing the name of a remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *name* points to an ASCIIZ string specifying the user name whose message forwarding is to be canceled.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_REM_NOT_LIST	51	This remote computer is not listening.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_NoNetworkResource	2105	A network resource shortage occurred.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.

Manifest	Value	Meaning
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_NetworkError	2136	A general network error has occurred.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NameNotFound	2273	The message alias cannot be found on the local area network.
NERR_AlreadyExists	2276	This message alias already exists locally.
NERR_TooManyNames	2277	The maximum number of added message aliases has been exceeded.
NERR_NameInUse	2283	The message alias is currently in use—try again later.
NERR_MsgNotStarted	2284	The messenger service has not been started.
NERR_NotLocalName	2285	The name is not on the local computer.
NERR_RemoteFull	2287	The message alias table on the remote station is full.
NERR_NameNotForwarded	2288	Messages for this alias are not currently forwarded.
NERR_DuplicateName	2297	A duplicate message alias exists on the local area network.
NERR_DeleteLater	2298	This message alias will be deleted later.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

NetMessageNameUnFwd requires that the *messenger* service be started.

Related Information

For information on:

- Forwarding a user's messages to another user—See "NetMessageNameFwd" on page 3-182.
- Listing the user name entries in a message name table—See "NetMessageNameEnum" on page 3-179.
- The *messenger* service—See "Service Category" on page 3-298.

Named Pipe Category

The functions in the Named Pipe category control interprocess communication (IPC) for named pipes.

For information on the following functions, see the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference.*

DosBufReset (*local, DOS*)

DosCallNmPipe (*local, DOS*)

DosClose (*local, DOS*)

DosConnectNmPipe (*local*)

DosDisConnectNmPipe (*local*)

DosDupHandle (*local, DOS*)

DosMakeNmPipe (*local*)

DosOpen (*local, DOS*)

DosPeekNmPipe (*local, DOS*)

DosQFHandState (*local, DOS*)

DosQHandType (*local, DOS*)

DosQNmPHandState (*local, DOS*)

DosQNmPipeInfo (*local, DOS*)

DosQNmPipeSemState (*local*)

DosRead (*local, DOS*)

DosReadAsync (*local*)

DosSetFHandState (*local, DOS*)

DosSetNmPHandState (*local, DOS*)

DosSetNmPipeSem (*local*)

DosTransactNmPipe (*local, DOS*)

DosWaitNmPipe (*local, DOS*)

DosWrite (*local, DOS*)

DosWriteAsync (*local*).

The functions in the Named Pipe category control interprocess communication (IPC) for named pipes. They are used with the OS2.H and NETCONS.H include files.

These functions are provided by the base operating system and supported by the OS/2 LAN Server accross the network.

Description

A *named pipe* is a bidirectional interprocess communication facility that allows two processes, either local or remote, to communicate with each other over the network. A process that creates a named pipe is known as a *server process*, and a process that establishes a connection to a named pipe is known as a *client process.*

To create an instance of a named pipe on the local computer, an application must call the DosMakeNmPipe function. This function specifies information that enables the server process to control the named pipe and allows client processes to access the named pipe. In order to create a named pipe, DosMakeNmPipe requires the following:

- The name chosen for the named pipe—the format is **\pipe***name* for a local named pipe.
- The directions (inbound, outbound, and full-duplex) that the named pipe can send and receive data.
- An indication as to whether the handle of the named pipe can be passed to spawned processes.
- The number of concurrent instances of the named pipe that can be created.
- The low-level parameters used by the OS/2 program.

An inbound or outbound named pipe (synonymous with *anonymous* pipes used with other multi-tasking operating systems) allows a process only to read or write by way of one handle. A full-duplex named pipe allows a process to both read and write data by way of one handle.

In some applications (especially those that were developed before the OS/2 Version 1.1 program or those that emulate anonymous pipes), the handle of a named pipe cannot be passed to a spawned process.

Each time DosMakeNmPipe is called with the same parameter information, another instance of the named pipe is created. Each instance is associated with a unique handle which is returned by DosMakeNmPipe. Thus, if DosMakeNmPipe is called five times with the same information, five different instances (or handles) of the same named pipe are created.

Other low-level operating system parameters such as the stream type and blocking mode of the named pipe can be set. Even though a server process creates a named pipe on a computer, client processes cannot access an instance of that named pipe until the server process calls DosConnectNmPipe. This function informs the system that a client process has permission to access an instance of the named pipe and also returns a handle to the named pipe.

To become a client process, an application must open an instance of the named pipe by calling the DosOpen function. DosOpen returns a handle to the client process that can be passed to other named pipe reading and writing functions. If DosOpen returns the ERROR_PIPE_BUSY error code (pipe currently being accessed by another process), the client process should call DosWaitNmPipe to wait for the named pipe to become available. DosWaitNmPipe can be configured to time out after a particular period of time, or to wait indefinitely for an instance of the named pipe.

When an instance of the named pipe becomes available, the DosWaitNmPipe function is allowed to return to the waiting client process. At this point, the client process can call DosOpen to open the named pipe. After a client process has opened an instance of a named pipe, the client process can begin to read and write to the named pipe. To perform these tasks, the client process should call the DosRead and DosWrite functions. Both of these functions accept the handle returned by DosOpen, and operate on the same thread of execution as the client process. If a server or client process requires that the reading and writing of named pipes be

executed on a separate thread, the process can call the DosReadAsync and DosWriteAsync functions. Note that a remote named pipe can be written to by specifying the pipe name as follows:

```
\\server\pipe\name
```

A process can also call the DosBufReset function to force all data to be written to a named pipe; normally, data written to a named pipe is held temporarily in a data buffer.

Note that output cannot be redirected to a named pipe.

Since a named pipe must be written to before it is read from, a process can call DosPeekNmPipe to see if there is any data written to a named pipe. DosPeekNmPipe reads the data in a named pipe but does not remove the data.

If necessary, either a server or client process can call DosDupHandle to replicate a handle to a named pipe. DosDupHandle returns a new handle to the same instance of a named pipe that an old handle represented. This handle can be passed to any named pipe function that could use the old handle.

Two functions are provided that decrease the overhead involved in writing to and reading from a named pipe. These two functions are DosTransactNmPipe and DosCallNmPipe. DosTransactNmPipe writes a message to and then reads a message from an opened named pipe. DosCallNmPipe opens, writes to, reads from, and then closes a named pipe. This four-in-one process is helpful when implementing a remote procedure call (RPC) on the network.

When a client process no longer requires access to a named pipe, the DosClose function can be called to close the named pipe.

When a server process no longer requires an instance of a named pipe, the server process calls DosDisconnectNmPipe to remove that instance of the named pipe by specifying its handle. If a client process is still accessing the named pipe, DosDisconnectNmPipe forces the client process off.

Five functions are provided that enable server or client processes to obtain information about a named pipe or its handle, as follows:

Function	Purpose
DosQFHandState	Determines whether the handle can be inherited and if write-behind is allowed.
DosQHandType	Returns the type of handle.
DosQNmPHandState	Returns the low-level parameters associated with a handle and the operating mode of the pipe; declares the instance count.
DosQNmPipeInfo	Returns the size of buffers and the number of instances currently available.
DosQNmPipeSemState	Returns the state of a semaphore associated with a named pipe.

The OS/2 LAN Requester/Server software provides three functions that enable server or client processes to set specific information about a named pipe that can be queried. The functions and settable parameters are as follows:

Function	Purpose
DosSetFHandState	Sets whether a handle of a named pipe can be inherited and if write-behind is allowed.
DosSetNmPHandState	Sets low-level parameters associated with pipes such as reading and writing mode.
DosSetNmPipeSem	Sets the association of a semaphore to a named pipe.

The transfer mode of a named pipe is set by either DosMakeNmPipe or DosSetNmPHandState; a named pipe transfers data in byte-stream or message-stream mode.

A named pipe operating in byte-stream mode operates like an anonymous pipe where all data written is transferred without any special processing performed on it. When operating in message-stream mode, a named pipe can distinguish between the different messages (and size of each message) read from and written to that named pipe.

Named pipes are designed so that a client process has no requirement for the type of resource it is opening (pipe or file). Client processes use the DosOpen, DosRead, and DosClose functions to open, read, and close both types of resources without reference to one resource being a file and the other a named pipe.

The following table describes the transition state of a named pipe, based on the action a server or client process indicates:

Current State	Action/Process	Next State
Pipe does not exist	DosMakeNmPipe, server	NP_DISCONNECTED
NP_DISCONNECTED	DosConnectNmPipe, server	NP_LISTENING
NP_LISTENING	DosOpen, client	NP_CONNECTED
NP_CONNECTED	DosDisconnectNmPipe, server	DISCONNECTED
NP_CONNECTED	DosClose, client	NP_CLOSING
NP_CLOSING	DosDisconnectNmPipe, server	DISCONNECTED
NP_CONNECTED	DosClose, server	NP_CLOSING

Note: The OS/2 DosChgFilePtr function (and other functions that perform seek operations on files) does not work with named pipes.

DOS Considerations

Under DOS, the functions can be executed only on a remote server that has interprocess communication (IPC) shares.

DOS supports only client processes; a pipe must have already been created and connected on a remote server. Child processes inherit the open file handles of a parent process. DOS does not support asynchronous reading and writing of named pipes.

Note: The Family API (FAPI) replacement library routine for DosOpen provides support for DASD opens (open Mode Flag 0x8000). Since DOS does not support this operation, pipe operations on this type of file handle will return ERROR_INVALID_HANDLE rather than ERROR_BAD_PIPE.

DosBufReset works differently depending on which version of DOS you are programming under. Under version 3.3 and 4.0, DosBufReset returns 0 after resetting a closed named pipe. If the handle is to a named pipe that has already been closed, DosBufReset returns ERROR_BROKEN_PIPE.

Under DOS 3.3 and 4.0, DosBufReset waits for the pipe to be emptied.

Related Information

For information on:

- Anonymous pipes, named pipes, or IPC—See the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference*, Volume 1.
- For a detailed description of each function—See the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference.*

Remote Utility Category

NetRemoteCopy (*local, DOS*)—See "NetRemoteCopy" on page 3-197.

NetRemoteExec (*local, server*)—See "NetRemoteExec" on page 3-200.

NetRemoteMove (*local, DOS*)—See "NetRemoteMove" on page 3-203.

NetRemoteTOD (*DOS*)—See "NetRemoteTOD" on page 3-206.

The functions in the Remote Utility category enable applications to copy and move remote files, remotely execute a program, and access the time-of-day information on a remote server. They are used with the REMUTIL.H and NETCONS.H include files.

Description

The NetRemoteCopy function performs optimized file copying. Files on a remote server are copied without physically moving the files to and from the local requester. The source and destination must be on the same server.

The NetRemoteMove function moves files or directories from one location to another on a remote server without physically moving the data if the source and destination are on the same drive. If source and destination are on different drives, the move does not require shuffling the data to and from the local requester.

To execute a program on a remote server, an application calls the NetRemoteExec function. NetRemoteExec performs the same tasks as the OS/2 DosExecPgm function, but on another network server.

The NetRemoteTOD function returns time-of-day information from a remote server.

DOS Considerations

Under DOS, the functions in the Remote Utility category enable applications to copy and move remote files and access the time-of-day information on a remote server. Attempting to execute the functions on a local requester returns NERR_RemoteOnly.

Data Structures

The function NetRemoteCopy uses data structure *copy_info*. The function NetRemoteMove uses data structure *move_info*. The function NetRemoteTOD uses data structure *time_of_day_info*. These data structures are described following the syntax descriptions in each function section.

NetRemoteExec does not use a data structure.

NetRemoteCopy

The NetRemoteCopy (local, DOS) function copies one or more files from one location to another on a remote server.

Syntax

```
#include <netcons.h>
#include <remutil.h>

unsigned far pascal
NetRemoteCopy (sourcepath, destpath, sourcepass, destpass,
               openflags, copyflags, buf, buflen)
const char far * sourcepath;
const char far * destpath;
const char far * sourcepass;
const char far * destpass;
unsigned short   openflags;
unsigned short   copyflags;
char far *       buf;
unsigned short   buflen;
```

where:

- *sourcepath* points to an ASCIIZ string containing the path name of the files to be copied (wildcards can be used). *sourcepath* must begin with either a redirected drive or a UNC name.
- *destpath* points to an ASCIIZ string containing the path name to which *sourcepath* is to be copied. For a wildcard *sourcepath*, *destpath* must be a directory. *destpath* must begin with either a redirected drive or a UNC name.
- *sourcepass* is reserved and must be NULL.
- *destpass* is reserved and must be NULL.
- *openflags* specifies how *destpath* will be opened. *openflags* is defined as follows:

Bit	Meaning
0-1	Used if *destpath* exists. If 0, the open fails. If 1, the file is appended. If 2, the file is overwritten.
2-3	Reserved, with a value of 0.
4	Used if *destpath* does not exist. If 0, the open fails. If 1, the file is created.
5-15	Reserved, with a value of 0.

- *copyflags* specifies how the file copy is done. *copyflags* is defined as follows:

Bit	Meaning
0	If 1, *destpath* must be a file. If bit 0 is set, bit 1 must be 0.
1	If 1, *destpath* must be a directory. If bit 1 is set, bit 0 must be 0.
2	If 0, *destpath* is opened in binary mode. If 1, *destpath* is opened in text mode.
3	If 0, *sourcepath* is opened in binary mode. If 1, *sourcepath* is opened in text mode.

Bit	Meaning
4	If 1, all writes are verified.
5-15	Reserved.

- *buf* points to the *copy_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Status of File Copy

NetRemoteCopy returns data in the following form:

```
struct copy_info {
   unsigned ci_num_copied;
   char     ci_err_buf[1];
};
```

where:

- *ci_num_copied* indicates the number of files that were copied.
- *ci_err_buf* is a variable-length ASCIIZ string containing error information pertaining to the file copy.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_FILE_NOT_FOUND	2	The file was not found.
ERROR_PATH_NOT_FOUND	3	The path was not found.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NO_MORE_FILES	18	No more files are available.
ERROR_SHARING_VIOLATION	32	A sharing violation occurred.
ERROR_FILE_EXISTS	80	The file already exists.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.

Manifest	Value	Meaning
NERR_BadSource	2381	The source path is illegal.
NERR_BadDest	2382	The destination path is illegal.
NERR_DifferentServers	2383	The source and destination paths are on different servers.

Other error return codes may be returned from the following OS/2 functions:

- DosFSCtl
- DosDevIOCtl(NIOCNETCOPY1_1).

Remarks

Currently, the source and destination for the file copy must be on the same server or an error results. The following cases are valid:

- The source and destination are both files. The source file is copied to the destination file, subject to *openflags* and *copyflags* limitations.
- The source is a file or wildcard and the destination is a directory. The source files are copied to the destination directory, subject to *openflags* and *copyflags* limitations.

Related Information

For information on:

- Listing the shared resources of a server—See "NetShareEnum" on page 3-350.
- Moving remote files between servers—See "NetRemoteMove" on page 3-203.
- Share passwords—See "Share Category" on page 3-337.

NetRemoteExec

The NetRemoteExec (local, server) function executes a program located on a remote server.

Syntax

```
#include <netcons.h>
#include <remutil.h>

unsigned far pascal
NetRemoteExec (reserved1, objnamebuf, objnamebufl, asyntraceflags,
               argpointer, envpointer, returncodes, pgmpointer,
               reserved2, remexecflags)
char far *       reserved1;
char far *       objnamebuf;
unsigned         objnamebufl;
unsigned         asyntraceflags;
const char far * argpointer;
const char far * envpointer;
char far *       returncodes;
const char far * pgmpointer;
char far *       reserved2;
unsigned short   remexecflags;
```

where:

- *reserved1* is a reserved pointer with the value of -1.
- *objnamebuf* points to the name of the object, such as a dynamic-link library. The NetRemoteExec function copies a name to this buffer if it could not successfully load and start the specified program.
- *objnamebufl* specifies the size (in bytes) of the *objnamebuf* memory area.
- *asyntraceflags* specifies the asynchronous and trace flags. *asyntraceflags* is defined as follows:

Value	Meaning
0	Synchronous process.
1	Asynchronous process without result code.
2	Asynchronous process with result code.

- *argpointer* points to a set of ASCIIZ strings containing the arguments of the file to be executed.
- *envpointer* points to a non-NULL ASCIIZ string specifying the environment for the file to be executed.
- *returncodes* points to an OS/2 data structure containing the return codes resulting from the file execution. This is the same data structure used with the OS/2 DosExecPgm function. For more information on the return codes and the DosExecPgm function, see the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference*, Volume 2.
- *pgmpointer* points to an ASCIIZ string containing only the name and extension of the file to be executed.
- *reserved2* is a reserved pointer with the value 0.

- *remexecflags* specifies the remote executable flags that control program execution. *remexecflags* is defined as follows:

Bit	Meaning	Manifest
0	REM_PIPE_MODE	If 0, a message mode pipe is used for standard input. If 1, a character mode pipe is used for standard input.
1	REM_WAIT_MODE	If 0, the OS/2 DosCWait function waits for the child process to finish before returning. If 1, DosCWait waits for all spawned processes to finish before returning.
2	REM_SIGL_MODE	If 0, map SIGINTR and SIGBREAK to SIGKILL when remoting standard signals. If 1, the send signals as received. For more information on OS/2 signals, see the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference*, Volume 1.
3-15		Reserved, with a value of 0.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
NERR_InternalError	2140	An internal error has occurred.
NERR_RunSrvPaused	2385	The run server you requested using the NET RUN command is paused.
NERR_ErrCommRunSrv	2389	An error occurred when communicating with a run server.
NERR_ErrConnRunSrv	2390	An error occurred when connecting to run server.
NERR_ErrorExecingGhost	2391	An error occurred when starting a background process.
NERR_ShareNotFound	2392	The shared resource you are connected to could not be found.

Manifest	Value	Meaning
NERR_PgmNotFound	2394	The program was not found.

Remarks

The NetRemoteExec function is a network extension of the OS/2 DosExecPgm function.

The executed process is run on the computer connected to the current drive of the caller. If the current drive of the caller is on a remote server, the child process is executed on that server. If the current drive of the caller is a local drive, the child process is executed locally.

The NetRemoteExec function requires that a remotely executed process inherit one of the following handles:

Handle	Meaning
0	Standard input (*stdin*)
1	Standard output (*stdout*)
2	Standard error (*stderr*)

When the NetRemoteExec function initiates an asynchronous process, the process identification (PID) returned to the first word of the *ReturnCodes* data structure is a valid local PID that represents the remote program. The PID can be passed to the OS/2 DosFlagProcess function to:

- Send signals to the remote process
- Call the OS/2 DosCWait function to wait for the remote process to exit
- Call the OS/2 DosKillProcess function to end the process.

Related Information

For information on:

- Listing resources of a server—See "NetShareEnum" on page 3-350.
- Listing the servers of a network—See "NetServerEnum2" on page 3-289.
- Executing a command on a server—See "NetServerAdminCommand" on page 3-284.
- Executing a program—See OS/2 DosExecPgm (*IBM Operating System/2 Technical Reference Version 1.2 Programming Reference*, Volume 1).
- OS/2 DosCWait—See OS/2 DosCWait (*IBM Operating System/2 Technical Reference Version 1.2 Programming Reference*, Volume 1).

NetRemoteMove

The NetRemoteMove (local, DOS) function moves one or more files from one location to another on a server.

Syntax

```
#include <netcons.h>
#include <remutil.h>

unsigned far pascal
NetRemoteMove(sourcepath, destpath, sourcepass, destpass,
              openflags, moveflags, buf, buflen)
const char far * sourcepath;
const char far * destpath;
const char far * sourcepass;
const char far * destpass;
unsigned         openflags;
unsigned         moveflags;
char far *       buf;
unsigned short   buflen;
```

where:

- *sourcepath* points to an ASCIIZ string containing the path name of the file to be moved (wildcards can be used). *sourcepath* must begin either with a redirected drive or a universal naming convention (UNC) name.
- *destpath* points to an ASCIIZ string containing the path name to which *sourcepath* is to be moved. For a wildcard *sourcepath*, *destpath* must be a directory. *destpath* must begin either with a redirected drive or a UNC name.
- *sourcepass* is reserved and must be NULL.
- *destpass* is reserved and must be NULL.
- *openflags* specifies how *destpath* will be opened. *openflags* is defined as follows:

Bit	Meaning
0-1	Used if *destpath* exists. If 0, the open fails. If 1, the file is appended. If 2, the file is overwritten.
2-3	Reserved, with a value of 0.
4	Used if *destpath* does not exist. If 0, the open fails. If 1, the file is created.
5-15	Reserved, with a value of 0.

- *moveflags* establishes control for the file move. *moveflags* is defined as follows:

Bit	Meaning
0	If 1, *destpath* must be a file, and bit 1 must be 0.
1	If 1, *destpath* must be a directory, and bit 0 must be 0.
2-15	Reserved; the value of these bits must be 0.

- *buf* points to the *move_info* data structure.

- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Status of File Move

The NetRemoteMove function returns data in the following form:

```
struct move_info {
   unsigned  mi_num_moved;
   char      mi_err_buf[1];
};
```

where:

- *mi_num_moved* indicates the number of files that were moved.
- *mi_err_buf* is a variable-length ASCIIZ string containing error information pertaining to the move operation.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_FILE_NOT_FOUND	2	The file was not found.
ERROR_PATH_NOT_FOUND	3	The path was not found.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NO_MORE_FILES	18	No more files are available.
ERROR_SHARING_VIOLATION	32	A sharing violation occurred.
ERROR_FILE_EXISTS	80	The file already exists.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_BadSource	2381	The source path is illegal.
NERR_BadDest	2382	The destination path is illegal.
NERR_DifferentServers	2383	The source and destination paths are on different servers.

Remarks

If the source and destination files are in the same directory, NetRemoteMove renames the source file. When the source and destination are on different drives, NetRemoteMove moves *sourcepath* to *destpath* and then deletes *sourcepath*.

Currently, the source and destination path names (*sourcepath* and *destpath*) supplied to the NetRemoteMove function must be on the same server. The following cases are valid:

- The source and destination are both files. The source file is copied to the destination file, subject to *openflags* and *moveflags* limitations.
- The source is a file or wildcard and the destination is a directory. The source files are copied to the destination directory, subject to *openflags* and *moveflags* limitations.

Related Information

For information on:

- Copying a file from one network location to another—See "NetRemoteCopy" on page 3-197.
- Determining if a driver letter is local or redirected to a remote server—See "NetUseGetInfo" on page 3-380.
- Listing available resources on a server—See "NetShareEnum" on page 3-350.

NetRemoteTOD

The NetRemoteTOD (DOS) function returns the time of day on a server.

Syntax

```
#include <netcons.h>
#include <remutil.h>

unsigned far pascal
NetRemoteTOD(servername, buf, buflen)
const char far * servername;
char far *       buf;
unsigned short   buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *buf* points to the *time_of_day_info* data structure.
- *buflen* specifies the size of the *buf* memory area.

Return Time-of-Day Information on a Server

The NetRemoteTOD function returns data in the following form:

```
struct time_of_day_info {
unsigned long   tod_elapsedt;
unsigned long   tod_msecs;
unsigned char   tod_hours;
unsigned char   tod_mins;
unsigned char   tod_secs;
unsigned char   tod_hunds;
unsigned short  tod_timezone;
unsigned short  tod_tinterval;
unsigned char   tod_day;
unsigned char   tod_month;
unsigned short  tod_year;
unsigned char   tod_weekday;
};
```

where:

- *tod_elapsedt* indicates the number of seconds that have elapsed since January 1, 1970.
- *tod_msecs* indicates the current millisecond.
- *tod_hours* indicates the current hour.
- *tod_mins* indicates the current minute.
- *tod_secs* indicates the current second.
- *tod_hunds* indicates the current hundredths of a second.
- *tod_timezone* indicates the timezone of the server; calculated (in minutes) from the greenwich mean time (GMT) zone.
- *tod_tinterval* indicates the time interval for each tick of the clock. Each integral integer represents 0.0001 second.

- *tod_day* (1 through 31) indicates the day of the month.
- *tod_month* (1 through 12) indicates the month.
- *tod_year* indicates the year, starting with 1980.
- *tod_weekday* indicates the day of the week (0 means Sunday, 6 means Saturday).

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Related Information

For information on listing the servers of a network, see "NetServerEnum2" on page 3-289.

Requester Category

NetWkstaGetInfo (*partially admin, DOS*)—See "NetWkstaGetInfo" on page 3-225.

NetWkstaSetInfo (*admin, DOS*)—See "NetWkstaSetInfo" on page 3-227.

NetWkstaSetUID2 (*DOS*)—See "NetWkstaSetUID2" on page 3-231.

The functions in the Requester category control the operation of requesters. They are used with the ACCESS.H, NETCONS.H, and WKSTA.H include files.

Description

The requester functions enable applications to:

- Log a user name onto a requester
- Log a user name off of a requester
- Execute a *logon script* on a logon server for a user name
- Control configuration of a requester.

To log a user name onto or off a requester or to execute a logon script for a user name, an application calls the NetWkstaSetUID2 function.

To configure a requester, an application calls the NetWkstaSetInfo function. The NetWkstaGetInfo function returns information about the configuration of a requester.

Note: The domain name of the requester (the *wki0_langroup* component in the *wksta_info* data structure) must not duplicate any computer name or user name on the network.

Giving a domain the name of a computer causes startup problems for one of two namesakes. If the like-named computer is started before the domain, no other computer in the domain will be able to start. And, conversely, if any computer in the like-named domain is started, the computer going by the domain name will be unable to start.

For more information on domains, see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*

DOS Considerations

Under DOS, the functions in the Requester category control the operation of requesters. They can be executed only on a local requester. Attempting to execute the functions on a remote server returns ERROR_NOT_SUPPORTED.

Certain parameters are not used under DOS and therefore cannot be set. However, validity checks are done on most of the unused parameters for future expansion.

The following table describes the parameters that are used and whether validity checks are perfomed on them:

Parameter	Used	Validity Checked
charwait	No	Yes
chartime	Yes	Yes
charcount	Yes	Yes
errlogsz	No	Yes
printbuftime	No	Yes
wrkheuristics	No	No

Data Structures

The NetWkstaGetInfo (level 0, 1, and 10) and NetWkstaSetInfo (level 0 and 1) functions accept or return data at the specified level of detail, using the *wksta_info* data structure.

Requester Information (Level 0)

```
struct wksta_info_0 {
  unsigned short wki0_reserved_1;
  unsigned long  wki0_reserved_2;
  char far *     wki0_root;
  char far *     wki0_computername;
  char far *     wki0_username;
  char far *     wki0_langroup;
  unsigned char  wki0_ver_major;
  unsigned char  wki0_ver_minor;
  unsigned long  wki0_reserved_3;
  unsigned short wki0_charwait;
  unsigned long  wki0_chartime;
  unsigned short wki0_charcount;
  unsigned short wki0_reserved_4;
  unsigned short wki0_reserved_5;
  unsigned short wki0_keepconn;
  unsigned short wki0_keepsearch;
  unsigned short wki0_maxthreads;
  unsigned short wki0_maxcmds;
  unsigned short wki0_reserved_6;
  unsigned short wki0_numworkbuf;
  unsigned short wki0_sizworkbuf;
  unsigned short wki0_maxwrkcache;
  unsigned short wki0_sesstimeout;
  unsigned short wki0_sizerror;
  unsigned short wki0_numalerts;
  unsigned short wki0_numservices;
  unsigned short wki0_errlogsz;
  unsigned short wki0_printbuftime;
  unsigned short wki0_numcharbuf;
  unsigned short wki0_sizcharbuf;
  char far *     wki0_logon_server;
  char far *     wki0_wrkheuristics;
  unsigned short wki0_mailslots;
};
```

where:

- *wki0_reserved_1, wki0_reserved_2, wki0_reserved_3, wki0_reserved_4, wki0_reserved_5,* and *wki0_reserved_6* are reserved, and must be 0.
- *wki0_root* points to an ASCIIZ string containing the path to the IBMLAN directory of the computer (the recommended default path is \IBMLAN).
- *wki0_computername* points to an ASCIIZ string containing the computer name of the local requester being configured.
- *wki0_username* points to an ASCIIZ string containing the user's name logged on the requester.
- *wki0_langroup* points to an ASCIIZ string containing the name of the domain to which the requester belongs.
- *wki0_ver_major* specifies the major version number of the OS/2 LAN Requester/Server software running on the computer.
- *wki0_ver_minor* specifies the minor version number of the OS/2 LAN Requester/Server software running on the computer.
- *wki0_charwait* indicates the number of seconds the requester will wait for a remote serial or communication device to become available.
- *wki0_chartime* indicates the number of milliseconds the requester will wait to collect data to send to a remote serial or communication device.
- *wki0_charcount* indicates the number of bytes of information the requester will send to a remote serial or communication device.
- *wki0_keepconn* indicates the number of seconds that an inactive connection from the requester to a resource of a server will be maintained.
- *wki0_keepsearch* indicates the number of seconds that an inactive search will continue.
- *wki0_maxthreads* indicates the number of threads the requester can dedicate to the network.
- *wki0_maxcmds* indicates the number of simultaneous network device driver commands that can be sent to the network.
- *wki0_numworkbuf* indicates the number of internal buffers the requester has.
- *wki0_sizworkbuf* indicates the size (in bytes) of each internal buffer.

  Use the following formula to determine the maximum values for both *numworkbuf* and the *sizworkbuf* parameters:

  `(sizworkbuf + 268) * numworkbuf ≤ 65,515`

  Default value: 4096

  Minimum value: 1024

  Maximum value: 16384

- *wki0_maxwrkcache* indicates the maximum size (in bytes) of an internal cache buffer.
- *wki0_sesstimeout* indicates the number of seconds that are waited before disconnecting an inactive session between a requester and a server.
- *wki0_sizerror* indicates the size (in bytes) of an internal error buffer.

- *wki0_numalerts* indicates the maximum number of clients that can receive alert messages.

  Note that each mailslot or semaphore registered by the NetAlertStart function is a different client, and that the alerter service registers at least three clients when it begins to run. For more information on alerts, see "Alert Category" on page 3-29.

- *wki0_numservices* indicates the number of services that can be started on the requester at any time. For more information on services, see "Service Category" on page 3-298.

- *wki0_errlogsz* indicates the maximum size (in kilobytes) of the error log file of the requester.

- *wki0_printbuftime* indicates the number of seconds waited before closing inactive compatibility-mode print jobs.

- *wki0_numcharbuf* indicates the number of character pipe buffers and device buffers the requester can have.

- *wki0_sizcharbuf* indicates the maximum size (in bytes) of a character pipe buffer and device buffer.

- *wki0_logon_server* points to an ASCIIZ string indicating the name of the domain controller of the requester. A NULL string indicates no logon servers are available. If the string is *, any available logon server is queried.

- *wki0_wrkheuristics* points to an ASCIIZ string of flags used to control a requester's operation. The heuristics default to values that are optimal for most configurations and normally need not be changed.

  The default value for *wki0_wrkheuristics* is defined as follows:

  ```
  wrkheuristics = 11111111213111111100010110201110
  ```

  The maximum value for *wki0_wrkheuristics* is defined in IBMLAN.INI as follows:

  ```
  wrkheuristics = 11221211223111111111519111213170
  ```

  If a partial string is specified, the default values are used for the remaining heuristics. If the string is NULL or is not present in the IBMLAN.INI file, the default string is used.

  If the values used are other than those listed (0 or 1 where no value is listed), errors may occur.

  The characters (from left to right) are defined as follows. Unless otherwise defined, 0 turns off a heuristic feature, and 1 turns on the feature.

  The parameters and their descriptions are as follows:

Parameter	Description
sizworkbuf	Sets the size of requester buffers, in bytes. Increase *sizworkbuf* to transfer large groups of data, such as database records, on the network. This value should be a multiple of 512. It should be the same for every requester on the network and equal to the *sizreqbuf* value used by servers. When the *numworkbuf* and *sizworkbuf* parameters are used in the following formula, they should not be greater than 64KB:

(sizworkbuf + 268) * numworkbuf ≤ 65515

Default value: 4096

Minimum value: 1024

Maximum value: 16384

wrkheuristics

Sets a variety of requester fine-tuning options. Each digit has an independent meaning. Missing digits are assumed to be the defaults as described. Except where noted, each is a binary digit where 0 means *off* or *inactive*, while 1 means *on* or *active*. The following are the meanings of the digits:

Digit	Meaning
0	Request opportunistic locking of files. The default is 1. When this heuristic is active, it allows a file opened in deny none sharing mode to be locked by the server (provided there are no other access requests), so that buffering can be used to enhance performance. The Server service assumes that the first requester is the only active process using that file and will prevent a second requester from accessing the file until buffer data is flushed (written to disk).
1	Do performance optimization for batch files. Heuristic 0 (opportunistic locking) must be set to 1. The default is 1. When this heuristic is active (set to 1), a batch file on the server executing on the requester is kept in the requester's buffer to prevent a request across the LAN for each line of the batch file. The batch file is opened and closed with each line executed; when this heuristic is inactive (set to 0), the close causes buffer data to be flushed.
2	Do asynchronous unlock and write unlock as follows: **Value** **Meaning** 0 Never 1 Always 2 Only on an OS/2 LAN Server virtual circuit. The default is 1. With this heuristic, files in the requester buffer are unlocked in the buffer, and processing continues without waiting for confirmation from the server. Any errors occurring at the server are reported later. Generally, the only errors that might occur are hard media errors, such as disk full or a loss of power to the server. (A virtual

circuit in this discussion is a NETBIOS™ session connection to another machine through a LAN.)

3 Do asynchronous close and write close as follows:

Value	Meaning
0	Never
1	Always
2	Only on an OS/2 LAN Server virtual circuit.

The default is 1.

This heuristic performs the same function in the close operation as *wrkheuristic* 2 performs in the unlock operation.

4 Buffer named pipes and serial devices. The default is 1.

This heuristic directs named pipes and communication devices through the requester's buffers.

5 Do combined lock read and write unlock as follows:

Value	Meaning
0	Never
1	Always
2	Only on an OS/2 LAN Server virtual circuit.

The default is 1.

6 Use open and read. The default is 1.

When this heuristic is active, a request to open a file also performs a read of size *sizworkbuf* from the beginning of the file to the requester's work buffer. This action anticipates that the data is subsequently read, saving an additional request across the LAN.

7 Reserved. This must be set to 1.

8 Use the chain send NETBIOS NCB as follows:

Value	Meaning
0	Never
1	Do if server's buffer is larger
2	Always (to avoid copy).

The default is 1.

When this heuristic is active, data packets larger than the LAN's transmit buffer size are chained together, eliminating some packet transmissions across the LAN.

NETBIOS is a trademark of International Business Machines Corporation.

9 Buffer small read and write requests (reading and writing a full buffer) as follows:

Value	Meaning
0	Never
1	Always
2	Only on an OS/2 LAN Server virtual circuit.

The default is 1.

When this heuristic is active and file access mode allows, requests to read or write data smaller than *sizworkbuf* are performed locally, in the requester's buffer, avoiding additional trips across the LAN. The buffer is flushed when the file is closed or when the buffer is needed to satisfy other requests.

This heuristic may enhance performance for applications that read, modify, and write back small records.

10 Use buffer mode (assuming shared access is granted) as follows:

Value	Meaning
0	Always read buffer size if request is smaller than buffer size.
1	Use full buffer if file is open for reading and writing.
2	Use full buffer if reading and writing sequentially.
3	Buffer all requests smaller than the buffer size.

The default is 3.

Shared access means the file was opened in sharing mode. These options allow selective tuning of the buffer mode if any applications handle data in a manner conflicting with buffering.

11 Use *raw* read and write server message block (SMB) protocols. The default is 1.

Raw read and write SMB protocols transfer data across the LAN without SMB headers. These protocols are used to transfer large files directly between a big buffer in the server and a work cache in the requester. When a large file transfer initiates *raw* read and write SMB protocols, the NETBIOS session involved exclusively uses the LAN adapter cards on both the send and receive stations until the request completes. Polling ensures large buffers are available before the transfer begins.

This heuristic may significantly improve performance of large file transfers across the LAN.

12 Use large *raw* read-ahead buffer. The default is 1.

This heuristic and heuristic 13 provide independent control over using raw SMB protocol for read-ahead and write-behind, respectively. Both are active with default values, but can be turned off to better suit a particular environment.

13 Use large *raw* write-behind buffer. The default is 1.

See digits 11 and 12 for more information.

14 Use *read multiplexing* server message block (SMB) protocols. The default is 1.

This SMB protocol is used for large read requests if large buffers described in heuristic 11 are unavailable, or *raw* SMB protocol is inactive. This protocol breaks transfers into buffer-size chunks (usually 4KB) and chains them together to satisfy the request. Exclusive use of LAN adapter cards does not occur.

15 Use *write multiplexing* SMB protocols. The default is 1.

This SMB protocol is used for large write requests if large buffers described in heuristic 11 are unavailable, or raw SMB protocol is inactive. This protocol divides transfers into buffer-size chunks (usually 4KB) and chains them together to satisfy the request. Exclusive use of LAN adapter cards does not occur.

16 Reserved. This must be set to 1.

17 Use same size small read-ahead or to sector boundary. The default is 1.

When this heuristic is active, requests to read small data records cause read-ahead in multiples of the data record size, so a full buffer is read and sent to the requester. Because multiple records may not fit evenly in the buffer, the last record in the buffer may be incomplete. However, no data is lost.

This heuristic is significant only if *wrkheuristic* 9 is inactive. The server will detect small data records of the same size being read sequentially and will establish the read-ahead operation.

18 Use same size small write-behind or to sector boundary. The default is 0.

When this heuristic is active, requests to write

small data records cause write-behind in multiples of the data record size, so a full buffer is written to the server. Because multiple records may not fit evenly in the buffer, the last record written may be incomplete. However, no data is lost.

This heuristic is significant only if *wrkheuristic* 9 is inactive. The server will detect small data records of the same size being written sequentially and will establish the write-behind operation.

19 Reserved. This must be set to 1.

20 Flush pipes and devices on DosBufReset or DosClose as follows:

Value	Meaning
0	Flush only files and devices opened by the caller. Spin until flushed (wait for confirmation before proceeding with other tasks).
1	Flush only files and devices opened by the caller. Flush only once.
2	Flush all files and all input and output of short-term pipes and devices. Spin until flushed.
3	Flush all files and all input and output of short-term pipes and devices. Flush only once.
4	Flush all files and all input and output of pipes and devices. Spin until flushed.
5	Flush all files and all input and output of pipes and devices. Flush only once.

The default is 0.

This heuristic gives the requester application more flexibility as to which files, pipes, or devices are flushed (written to disk) when DosBufReset or DosClose is done.

21 Used to support OS/2 LAN Server encryption. The default is 1.

22 Control log entries for multiple occurrences of an error. A recurring error can fill up the error log; use this heuristic to keep down the number of log entries. If the value is other than 0, the first, fourth, eighth, 16th, and 32nd occurrences of an error are logged. After that, every 32nd further occurrence is logged.

If the value is other than 0, it also defines the size of an error table. The table is a record of what errors have occurred. If an error does not

match an existing entry in the table, it replaces the entry with the lowest number of occurrences.

Set the value as follows:

Value	Meaning
0	Log all occurrences
1	Use error table, size 1
2	Use error table, size 2
3	Use error table, size 3
4	Use error table, size 4
5	Use error table, size 5
6	Use error table, size 6
7	Use error table, size 7
8	Use error table, size 8
9	Use error table, size 9.

The default is 0.

23 Buffer all files opened with deny-write sharing mode. The default is 1.

When this heuristic is active, the server buffers all files opened with deny-write sharing mode, even if the access mode is not read-only.

This heuristic deactivates buffering on this requester if an application does not work correctly with it.

24 Buffer all files opened with read only access. The default is 1.

When this heuristic is active, the server buffers all files opened with read-only access mode, even if the sharing mode is not deny-write.

This heuristic deactivates buffering on this requester if an application does not work correctly with it.

25 Read ahead when opening for execution. Reading an executable file sequentially is usually, but not always, faster. The default is 1.

This heuristic value should be 1 for many executable files loaded across the LAN. For example, DisplayWrite™ 4/2 load time decreases by more than 50 percent. Experiment with your particular program to determine which option is better.

26 Handle Control-C (Ctrl+C) as follows:

Value	Meaning
0	No interrupts allowed
1	Only allow interrupts on long-term operations
2	Always allow interrupts.

The default is 2.

27 Force correct open mode when creating files on a core server. (A core server is a DOS-based LAN server, such as PC LAN Program 1.3.) OS/2 LAN Server does not allow DOS-based servers. The default is 0.

28 Use the NETBIOS *NoAck* mode (transferring data without waiting for an acknowledgement) as follows:

Value	Meaning
0	NoAck is never used (disable NoAck)
1	NoAck on send only.

The default is 1.

29 Send data along with the server message block write block *raw* requests. This may save time. The default is 1.

When this heuristic is active, the requester sends a requester buffer of data to the server with its request for big buffers to use for large file transfers. This action may save time if the server has limited big buffers (*numbigbufs*) compared to the number of requesters trying to send large files.

DisplayWrite is a trademark of International Business Machines Corporation.

30 Send a popup message to the screen when the requester logs an error, as follows:

Value	Meaning
0	Never
1	On write fault errors only (no timeout)
2	On write fault and internal errors only (no timeout)
3	On all errors (no timeout)
4	Reserved.
5	On write fault errors only (timeout)
6	On write fault and internal errors only (timeout)
7	On all errors (timeout).

The default is 1.

Values other than 1 are normally used for debug purposes only.

31 Reserved.

wrknets Lists names of networks the requester runs on. Names of available networks are listed in the Networks section of the IBMLAN.INI file. The OS/2 LAN Server supports only a single network.

Required value: net1

wrkservices Specifies network services to start with the Requester service. For example, the Messenger service, which sends and receives network messages, can be started with the Requester service. The options are **Messenger** and **Netpopup**. This value is defined by the user at installation.

- *wki0_mailslots* specifies the maximum number of mailslots allowed.

Requester Information (Level 1)

Requester information level 1 includes all the fields of *wksta_info_0*, plus *oth_domains* and *logon_domains*.

```
struct wksta_info_1 {
  unsigned short wki1_reserved_1;
  unsigned long  wki1_reserved_2;
  char far *     wki1_root;
  char far *     wki1_computername;
  char far *     wki1_username;
  char far *     wki1_langroup;
  unsigned char  wki1_ver_major;
  unsigned char  wki1_ver_minor;
  unsigned long  wki1_reserved_3;
  unsigned short wki1_charwait;
  unsigned long  wki1_chartime;
  unsigned short wki1_charcount;
  unsigned short wki1_reserved_4;
  unsigned short wki1_reserved_5;
  unsigned short wki1_keepconn;
  unsigned short wki1_keepsearch;
  unsigned short wki1_maxthreads;
  unsigned short wki1_maxcmds;
  unsigned short wki1_reserved_6;
  unsigned short wki1_numworkbuf;
  unsigned short wki1_sizworkbuf;
  unsigned short wki1_maxwrkcache;
  unsigned short wki1_sesstimeout;
  unsigned short wki1_sizerror;
  unsigned short wki1_numalerts;
  unsigned short wki1_numservices;
  unsigned short wki1_errlogsz;
  unsigned short wki1_printbuftime;
  unsigned short wki1_numcharbuf;
  unsigned short wki1_sizcharbuf;
  char far *     wki1_logon_server;
  char far *     wki1_wrkheuristics;
  unsigned short wki1_mailslots;
  char far *     wki1_logon_domain;
  char far *     wki1_oth_domains;
  unsigned short wki1_numdgrambuf;
};
```

where:

- *wki1_reserved_1*, *wki1_reserved_2*, *wki1_reserved_3*, *wki1_reserved_4*, *wki1_reserved_5*, and *wki1_reserved_6* are reserved, and must be 0.
- *wki1_root* points to an ASCIIZ string containing the path to the IBMLAN directory (the recommended default path is IBMLAN\) of a computer .
- *wki1_computername* points to an ASCIIZ string containing the computer name of the local requester being configured.
- *wki1_username* points to an ASCIIZ string containing the user's name logged on the requester.
- *wki1_langroup* points to an ASCIIZ string containing the name of the domain to which the requester belongs.

- *wki1_ver_major* specifies the major version number of the OS/2 LAN Requester/Server software running on the computer.
- *wki1_ver_minor* specifies the minor version number of the OS/2 LAN Requester/Server software running on the computer.
- *wki1_charwait* indicates the number of seconds the requester will wait for a remote serial device resource to become available.
- *wki1_chartime* indicates the number of milliseconds the requester will wait to collect data to send to a remote serial device resource.
- *wki1_charcount* indicates the number of bytes of information the requester will send to a remote serial device resource.
- *wki1_keepconn* indicates the number of seconds an inactive connection from the requester to the resource a server will be maintained.
- *wki1_keepsearch* indicates the number of seconds an inactive search will continue.
- *wki1_maxthreads* indicates the number of threads the requester can dedicate to the network.
- *wki1_maxcmds* indicates the number of simultaneous network device driver commands that can be sent to the network.
- *wki1_numworkbuf* indicates the number of internal buffers the requester has.
- *wki1_sizworkbuf* indicates the size (in bytes) of each internal buffer.
- *wki1_maxwrkcache* indicates the maximum size (in bytes) of an internal cache buffer.
- *wki1_sesstimeout* indicates the number of seconds that are waited before disconnecting an inactive session between a requester and a server.
- *wki1_sizerror* indicates the size (in bytes) of an internal error buffer.
- *wki1_numalerts* indicates the maximum number of clients that can receive alert messages.

  Note that each mailslot or semaphore registered by the NetAlertStart function is a different client, and that the alerter service registers at least three clients when it begins to run. For more information on alerts, see "Alert Category" on page 3-29.
- *wki1_numservices* indicates the number of services that can be started on the requester at any time. For more information on services, see "Service Category" on page 3-298.
- *wki1_errlogsz* indicates the maximum size (in kilobytes) of the error log file of a requester.
- *wki1_printbuftime* indicates the number of seconds that are waited before closing inactive compatibility-mode print jobs.
- *wki1_numcharbuf* indicates the number of character pipe buffers and device buffers the requester can have.
- *wki1_sizcharbuf* indicates the maximum size (in bytes) of a character pipe buffer and device buffer.

- *wki1_logon_server* points to an ASCIIZ string indicating the name of the Domain Controller of a requester. A NULL string indicates no logon servers are available. If the string is *, any available logon server is queried.
- *wki1_wrkheuristics* points to an ASCIIZ string of flags used to control a requester operation. The heuristics default to values that are optimal for most configurations and normally need not be changed. For character positions and meanings, refer to "Requester Information (Level 0)."
- *wki1_mailslots* specifies the maximum number of mailslots allowed.
- *wki1_logon_domains* names the domain to which the user is logged on. It is returned as NULL when no one is logged on.
- *wki1_oth_domains* field is an ASCIIZ string listing all domains on which the machine is currently enlisted. This is a *far* pointer to an ASCIIZ string which is a space-delimited list of domains. The *oth_domains* field is settable with NetWkstaSetInfo.
- *wkil_numdgrambuf* is the number of buffers allocated for receiving datagrams.

Requester Information (Level 10)

The *wksta_info_10* data structure is supplied to fulfill the needs of remote users who want to obtain certain information from a server. This data structure allows remote users to discover what domain a server belongs to.

Since a remote NetWkstaGetInfo at levels 0 and 1 requires administrative privilege, a remote user who does not have privilege level ADMIN cannot use those structures. This new level provides the needed information.

```
struct wksta_info_10 {
  char far *      wki10_computername;
  char far *      wki10_username;
  char far *      wki10_langroup;
  unsigned char   wki10_ver_major;
  unsigned char   wki10_ver_minor;
  char far *      wki10_logon_domain;
  char far *      wki10_oth_domains;
};
```

where:

- *wki10_computername* points to an ASCIIZ string containing the computer name of the requester being queried.
- *wki10_username* points to an ASCIIZ string containing the user's name logged on the requester.
- *wki10_langroup* points to an ASCIIZ string containing the name of the domain to which the requester belongs.
- *wki10_ver_major* specifies the major version number of the OS/2 LAN Requester/Server software running on the computer.
- *wki10_ver_minor* specifies the minor version number of the OS/2 LAN Requester/Server software running on the computer.
- *wki10_logon_domain* names the domain that the user is logged on to. It is returned as NULL when no one is logged on.
- *wki10_oth_domains* field is an ASCIIZ string listing all domains on which the machine is currently enlisted. This is a *far* pointer to an ASCIIZ string which is a space-delimited list of domains.

DOS Considerations

For information on the DOS LAN Requester heuristics, see the *DOS LAN Requester User's Guide*.

Related Information

For information on:

- Configuring requesters—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- The IBMLAN.INI file—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- Domains—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- NCB architecture—See the *IBM PC LAN Program 1.3 Application Programmer's Guide.*
- *wksta_info_0* components—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*

NetWkstaGetInfo

The NetWkstaGetInfo (partially admin, DOS) function returns information about the configuration components of a requester.

Syntax

```
#include <netcons.h>
#include <wksta.h>

unsigned far pascal
NetWkstaGetInfo(servername, level, buf, buflen, totalavail)
char far *           servername;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail (0, 1 or 10) to be returned in the *wksta_info* data structure.
- *buf* points to the *wksta_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

The ordinary user can get information from the level 10 data structure.

Related Information

For information on modifying the configuration of a local requester, see "NetWkstaSetInfo" on page 3-227.

NetWkstaSetInfo

The NetWkstaSetInfo (admin, DOS) function configures a requester.

Syntax

```
#include <netcons.h>
#include <wksta.h>

unsigned far pascal
NetWkstaSetInfo(servername, level, buffer, buflen, parmnum)
char far *     servername;
short          level;
char far *     buf;
unsigned short buflen;
short          parmnum;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail (0 or 1) returned in the *wksta_info* data structure.
- *buf* points to the *wksta_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* determines whether *buf* contains a complete *wksta_info* data structure or a single data structure component. If *parmnum* is 0, *buf* must contain a complete *wksta_info* data structure. Otherwise, *parmnum* must specify the ordinal position value for one of the following *wksta_info* data structure components, as defined as follows in WKSTA.H:

Manifest	Value	Component
WKSTA_CHARWAIT_PARMNUM	10	*wkix_charwait*
WKSTA_CHARTIME_PARMNUM	11	*wkix_chartime*
WKSTA_CHARCOUNT_PARMNUM	12	*wkix_charcount*
WKSTA_ERRLOGSZ_PARMNUM	27	*wkix_errlogsz*
WKSTA_PRINTBUFTIME_PARMNUM	28	*wkix_printbuftime*
WKSTA_WRKHEURISTICS_PARMNUM	32	*wkix_wrkheuristics*
WKSTA_OTHDOMAINS_PARMNUM	35	*wkix_oth_domains*

Note: x = 0 or 1.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_REM_NOT_LIST	51	This remote computer is not listening.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_NoNetworkResource	2105	A network resource shortage occurred.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.

Manifest	Value	Meaning
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_NameNotFound	2273	The message alias cannot be found on the local area network.
NERR_AlreadyExists	2276	This message alias already exists locally.
NERR_TooManyNames	2277	The maximum number of added message aliases has been exceeded.
NERR_DuplicateName	2297	A duplicate message alias exists on the local area network.
NERR_DeleteLater	2298	This message alias will be deleted later.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosDevIOCtl
- DosFSCtl
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- redir.GetBiosInfo[-ERROR_NO_MORE_FILES]
- redir.NIOCBiosOpen.

Remarks

The fields *wki0_computername*, *wki0_langroup*, are not settable by users or administrators.

The values of the field *oth_domains* in the *wksta_info* data structure are separated by spaces. An empty list is legal. As usual a NULL pointer means "do not modify this field." An empty element is not legal. When setting *oth_domains*, the API will reject the request in the following cases:

- The name list was incorrect
- One of the names could not be added to the network adapters managed by OS/2 LAN Requester/Server.

Related Information

For information on retrieving the comfiguration of a local requester, see "NetWkstaGetInfo" on page 3-225.

NetWkstaSetUID2

The NetWkstaSetUID2 (DOS) function registers a user name and password with the requester and validates the user account. It returns a structure with information about the logon. This function takes an optional domain name argument. If it is absent, it defaults to the primary domain (*wki0_langroup*) of the requester.

Syntax

```
#include <netcons.h>
#include <wksta.h>
#include <access.h>

unsigned far pascal
NetWkstaSetUID2(reserved,domainname, username, password, parms,
                ucond, level, buf, buflen, totalavail)
char far *          reserved;
char far *          domainname;
char far *          username;
char far *          password;
char far *          parms;
unsigned short      ucond;
short               level;
char far *          buf;
unsigned short      buflen;
unsigned short far * totalavail;
```

where:

- *reserved* is a reserved parameter, it must be NULL.
- *domainname* is the name of the domain to log on to. If this parameter is NULL or if it points to a NULL string, the primary domain of the requester is used.
- *username* points to an ASCIIZ string containing the user name to be logged onto the requester. Specifying a NULL string logs the user name off the requester.
- *password* points to an ASCIIZ string containing the password of the user name, obtained by an application's request to the user. A NULL pointer or string indicates no password is needed. *password* becomes the default password for requester and is used whenever the requester attempts to access a remote resource.
- *parms* must be NULL and is reserved.
- *ucond* specifies what action to take if another user name is logged on the requester. The WKSTA.H include file defines four values:

Manifest	Value	Meaning
WKSTA_NOFORCE	0	NetWkstaSetUID2 fails, and the user's identification number (UID) does not change.
WKSTA_FORCE	1	Logs the current user name off, disconnecting any connections to redirected resources.

Manifest	Value	Meaning
WKSTA_LOTS_OF_FORCE	2	Cancels any connections and other pending activities necessary. Fails if any connection is used by a process as the current drive.
WKSTA_MAX_FORCE	3	Always succeeds—forces all disconnections.

- *level* level of data structure to return. It must be 1.
- *buf* is the pointer to the buffer *user_logon_info_1* or *user_logoff_info_1* for return data. See the data structures under "User Category" on page 3-382.
- *buflen* is the length of the buffer.
- *totalavail* is the total information available on return.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
VALIDATED_LOGON	1	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_REM_NOT_LIST	51	This remote computer is not listening.
ERROR_INVALID_PASSWORD	86	The specified password is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_NoNetworkResource	2105	A network resource shortage occurred.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_NetworkError	2136	A general network error has occurred.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_AlreadyLoggedOn	2200	This workstation is already logged on.
NERR_NotLoggedOn	2201	This workstation has not been logged on yet.
NERR_BadUsername	2202	The user name or group name parameter is invalid.
NERR_BadPassword	2203	The password parameter is invalid.
NERR_UnableToAddName_W	2204	The logon processor did not add the message alias.

Manifest	Value	Meaning
NERR_UnableToAddName_F	2205	The logon processor did not add the message alias.
NERR_UnableToDelName_W	2206	The logoff processor did not delete the message alias.
NERR_UnableToDelName_F	2207	The logoff processor did not delete the message alias.
NERR_LogonsPaused	2209	The network logons are paused.
NERR_LogonDomainExists	2216	There is already a logon domain for this computer.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_PasswordTooShort	2245	The password is shorter than required.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_NameNotFound	2273	The message alias cannot be found on the local area network.
NERR_AlreadyExists	2276	This message alias already exists locally.
NERR_TooManyNames	2277	The maximum number of added message aliases has been exceeded.
NERR_DuplicateName	2297	A duplicate message alias exists on the local area network.
NERR_DeleteLater	2298	This message alias will be deleted later.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Manifest	Value	Meaning
NERR_ActiveConns	2402	Active connections still exist.
NERR_LastAdmin	2452	The last administrator cannot be deleted.
NERR_LogonTrackingError	2454	Unable to set logon information for this user.
NERR_NetLogonNotStarted	2455	The Netlogon service has not been started.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NIOCGETUSERNAME)
- DosFsCtl(NIOCSETUSERNAME)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize[-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite
- int2F[2F_NetSetUserName]/*DOS only*/
- redir.GetBiosInfo[-ERROR_NO_MORE_FILES]
- redir.NIOCBiosOpen
- redir.NIOCGetUserName
- redir.NIOCSetUserName.

Remarks

The NetWkstaSetUID2 function performs a range of duties, assisted by internal calls. It returns a structure with information about the logon (user_logon_info_1) or the logoff (user_logoff_info_1).

If a user name is already logged onto the requester (UID already in effect), NetWkstaSetUID2 takes the action specified in *ucond*—either failing or forcing the user to log off.

A NULL user name is interpreted as a network logoff.

If the user's logon is invalid, the API returns ERROR_ACCESS_DENIED. The *usrlog1_code* code field in the *user_logon_info_1* data structure is valid even when the API returns ERROR_ACCESS_DENIED. The other fields are valid only when the return code is Validated Logon.

The values of the *usrlog1code* fields can be as follows:

Manifest	Meaning
NERR_Success	No errors were encountered.
NERR_PasswordExpired	The user has an account, but the user's password has expired. No other conditions of logon have been checked.
NERR_InvalidWorkstation	The user was attempting to log on from an invalid requester.
NERR_InvalidLogonHours	The user was attempting to log on at an invalid time.
ERROR_ACCESS_DENIED	Some condition of logon has not been met.
NERR_StandaloneLogon	No domain controller could be found to validate the user. Script processing was not performed.
NERR_NonValidatedLogon	The logon server could not validate the logon request.
NERR_LogonScriptError	An error occurred while processing logon script.

The following table defines the other fields that are valid in the *user_logon_info_1* data structure for each code listed in the previous table:

Error Returned from NetWkstaSetUID2	Manifest Logon	Valid Fields
NERR_Success	NERR_Standalone	None
NERR_UnableToAddName_W	NERR_NonValidatedLogon	Computer name, Script path
	NERR_Success	All

Error Returned from NetWkstaSetUID2	Manifest Logon	Valid Fields
ERROR_ACCESS_DENIED	NERR_PasswordExpired	None
	NERR_InvalidWorkstation	None
	NERR_InvalidLogonHours	None
	NERR_LogonScriptError	None
	ERROR_ACCESS_DENIED(**)	None
All other errors	None. Code is meaningless	None

(**) For no account, account disable, and account expired, the password does not match.

The following table defines fields for logging off:

Error Returned from NetWkstaSetUID2	Manifest Logon	Valid Fields
NERR_Success	NERR_StandaloneLogon	None
NERR_UnableToDelName_W	NERR_NonValidatedLogon	None
	NERR_Success	All
All other errors	None. Code is meaningless	None

The *ucond* parameter has meaning only when the API is called to log someone off. These meanings are as follows:

Check all outstanding conditions	ucond = 0	ucond = 1	ucond = 2	ucond = 3
1)Any drive current	Fail. Do no disconnects.	Fail. Do no disconnects.	Fail. Do no disconnects.	Force disconnect succeed.
2) Any drive	Fail. Do no disconnects.	Fail. Do no disconnects.	Force closed, succeed.	Force closed, succeed.
3) Any in use (net use..)	Fail. Do no disconnects.	Disconnect, Unuse, succeed.	Disconnect, Unuse, succeed.	Disconnect, Unuse, succeed.
4) Anything dormant	Disconnect and succeed.	Disconnect and succeed.	Disconnect and succeed.	Disconnect and succeed.

Related Information

For information on configuring the local requester, see "NetWkstaSetInfo" on page 3-227.

Serial Device Category

NetCharDevControl (*admin, server, DOS*)—See "NetCharDevControl" on page 3-243.

NetCharDevEnum (*admin, server, DOS*)—See "NetCharDevEnum" on page 3-246.

NetCharDevGetInfo (*server, DOS*)—See "NetCharDevGetInfo" on page 3-248.

NetCharDevQEnum (*server, DOS*)—See "NetCharDevQEnum" on page 3-251.

NetCharDevQGetInfo (*server, DOS*)—See "NetCharDevQGetInfo" on page 3-254.

NetCharDevQPurge (*admin, server, DOS*)—See "NetCharDevQPurge" on page 3-257.

NetCharDevQPurgeSelf (*server, DOS*)—See "NetCharDevQPurgeSelf" on page 3-260.

NetCharDevQSetInfo (*admin, server, DOS*)—See "NetCharDevQSetInfo" on page 3-263.

The functions in the Serial Device category control shared serial devices and their associated queues. They are used with the CHARDEV.H and NETCONS.H include files.

Description

In order for an application to communicate with a device such as a serial printer, the application must be able to communicate directly and interactively. The communication must allow commands to be submitted dynamically and protocols to be changed as the application executes. The OS/2 LAN Requester/Server software defines these types of communication devices as *serial devices*. This definition is not limited to devices attached to a hardware serial ports.

The OS/2 LAN Requester/Server software can pool serial devices of the same type into a serial device queue to which a requesting application makes its connection. A serial device queue can contain one or more serial devices and simultaneously allow multiple applications to individually connect to one of the available serial devices. Serial device queues can pool serial devices only on a server.

Before an application can communicate with a serial device, the following must occur:

- The server must have a serial device connected to one of its available LPT or COM ports.
- A serial device queue must be created and shared on the network.
- A requesting application must explicitly redirect a local or NULL device name to the shared serial device queue by calling NetUseAdd, or implicitly open the serial device queue by calling DosOpen.

Note: Serial device queues exist only while they are being shared. In contrast, a spooled device queue (such as for a printer) exists from the time it is created by calling the appropriate Add function to the time it is removed.

An application can explicitly redirect a local device name in order to connect to a serial device queue by calling the NetUseAdd function, or implicitly connect by calling the DosOpen function and specifying the name of a queue. An explicit

connection allows the application to refer to the serial device queue with a local device name. Note however, that an explicit connection does not open a serial device. For more information about redirecting a local device name to a shared resource, see "Use Category" on page 3-368.

To illustrate how serial devices and queues work on the LAN, consider the following scenario. Assume that there are four serial devices connected to the communication ports of a server in the following manner:

Port	Device
COM1, COM2, COM3	Printers
COM4	Special

Once the serial devices are connected to the ports of a servers, an application can create a serial device queue by calling the NetShareAdd function and specifying the share type as a serial device queue. In this scenario, assume the NetShareAdd function is called three times, creating the following three queues:

Queue name	Priority	Device name
SPLQ	5	COM4
PRINTQ	5	COM1, COM2, COM3
VIPPRINT	1	COM3

Note that the COM3 port is allocated for use by two different queues, *PRINTQ* and *VIPPRINT*. After the NetShareAdd function is called to create a queue, a parameter can be set assigning a priority to the queue by calling NetCharDevQSetInfo. The priority can be in the range from 1 (high) through 9 (low). Generally, the OS/2 LAN Requester/Server software allows requests to serial device queues with a higher priority to access the pool of serial devices before other queues with lower priorities.

At this point, an application can connect to the shared serial device queue and begin communicating with one of the pooled serial devices in the queue.

If there happens to be more than one available serial device in a serial device queue, the queue returns the first available serial device to the requesting application. If no devices are currently available, the queue puts the request on a waiting list until a serial device becomes available. Note that the queue will wait only as long as the *charwait* parameter of a requester specifies. If the thread undergoes a time out while waiting for a serial device to become available, the DosOpen function returns the ERROR_BAD_NET_RESP error code.

An application can determine if a particular serial device is working by calling the NetCharDevGetInfo function or check all devices by calling the NetCharDevEnum function. An application can also check to see if the queue is busy or where the request of the application is on the queue waiting list by calling the NetCharDevQGetInfo function. To check all queues, call the NetCharDevQEnum function.

Applications can call the NetCharDevQPurgeSelf function to eliminate all requests submitted to a particular serial device queue from the requester of that application. Or, all requests submitted by all applications can be removed from the queue by

calling NetCharDevQPurge. Note that a process that currently has a device open is unaffected.

When the application no longer needs the device, it should call the DosClose function to return control of the serial device to the serial device queue, allowing it to be used by another application. If, for some reason, the application cannot successfully call the DosClose function to close the serial device queue, the NetCharDevControl function can be called to force the serial device queue closed.

Note that when an application successfully opens a remote serial device, the values of the *chartime* and *charcount* components from the *wksta_info* data structure are used to control how information flows across the network to other pertinent requesters and servers. Any application modifying these values on the requester where the open was performed should note the following:

- *chartime* and *charcount* affect all applications running on the requester
- Network efficiency, network response time, and network throughput may be slowed.

For more information on requester parameters, see "Requester Category" on page 3-208.

Data Structures

The *level* parameter controls the level of information provided to or returned from the data structures used by the NetCharDevEnum, NetCharDevGetInfo, NetCharDevQEnum, NetCharDevQGetInfo, and NetCharDevQSetInfo functions.

Serial Device Information (Level 0)

The NetCharDevEnum and NetCharDevGetInfo functions use the following data structure when the *level* parameter is 0:

```
struct chardev_info_0 {
   char ch0_dev[DEVLEN+1];
};
```

where:

- *ch0_dev* is an ASCIIZ string containing the device name associated with the serial device.

Serial Device Queues Information (Level 0)

The NetCharDevQEnum, NetCharDevQGetInfo, and NetCharDevQSetInfo functions use the following data structure when the *level* parameter is 0:

```
struct chardevQ_info_0 {
   char cq0_dev[NNLEN+1];
};
```

where:

- *cq0_dev* is an ASCIIZ string containing the queue name for the serial device queue.

Serial Device Information (Level 1)

The NetCharDevEnum and NetCharDevGetInfo functions use the following data structure when the *level* parameter is 1:

```
struct chardev_info_1 {
  char          ch1_dev[DEVLEN+1];
  char          ch1_pad1;
  unsigned short ch1_status;
  char          ch1_username[UNLEN+1];
  char          ch1_pad2;
  unsigned long  ch1_time;
};
```

where:

- *ch1_dev* is an ASCIIZ string specifying the devicename associated with the serial device.
- *ch1_pad1* WORD-aligns the data structure components.
- *ch1_status* specifies the status of the device. *ch1_status* is defined as follows:

Bit	Meaning
0	Reserved, with a value of 0.
1	If 0, the device is idle; if 1, the device is opened, and presumably in use by some application.
2	If 0, the device has encountered no errors; if 1, the device has encountered an error.
3-15	Reserved, with a value of 0.

- *ch1_username* is an ASCIIZ string specifying the current user of the device.
- *ch1_pad2* WORD-aligns the data structure components.
- *ch1_time* specifies the number of seconds the current application has been connected to the serial device.

Serial Device Queues Information (Level 1)

The NetCharDevQEnum, NetCharDevQGetInfo, and NetCharDevQSetInfo functions use the following data structure when the *level* parameter is 1:

```
struct chardevQ_info_1 {
  char          cq1_dev[NNLEN+1];
  char          cq1_pad;
  unsigned short cq1_priority;
  char far *     cq1_devs;
  unsigned short cq1_numusers;
  unsigned short cq1_numahead;
};
```

where:

- *cq1_dev* is an ASCIIZ string specifying the queue name.
- *cq1_pad* WORD-aligns the data structure components.

- *cql_priority* specifies the queue priority. *cql_priority* can be in the range from 1 through 9, where 1 has highest priority.
- *cql_devs* points to an ASCIIZ string containing the device names assigned to the queue (such as `COM1 COM3`).
- *cql_numusers* specifies the number of user names in the queue.
- *cql_numahead* specifies the number of user names in front of a particular user. To find the number of users, specify *username* with the NetCharDevQEnum or NetCharDEvQGetInfo function. If *cql_numahead* is -1, then *username* is not currently in the queue.

NetCharDevControl

The NetCharDevControl (admin, server, DOS) function forces a serial device closed.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetCharDevControl(servername, devname, opcode)
char far * servername;
char far * devname;
short      opcode;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *devname* points to an ASCIIZ string specifying the device to modify.
- *opcode* specifies how to modify *devname*. *opcode* is defined in CHARDEV.H as follows:

Manifest	Value	Meaning
CHARDEV_CLOSE	0	Closes the serial device

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_SEM_TIMEOUT	121	A time out happened from the semaphore API functions.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.

Manifest	Value	Meaning
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_DevInvalidOpCode	2331	The operation is invalid for this device.
NERR_DevNotFound	2332	This device cannot be shared.
NERR_DevNotOpen	2333	This device was not open.
NERR_NoCommDevs	2337	There are no shared communication devices.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFSRamSemClear
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- DosSemRequest.

Remarks

Normally, a serial device is closed with a call to the DosClose function. If for some reason the process that opened the device cannot close it, NetCharDevControl can be called to force the device closed.

Related Information

For information on:

- Modifying a serial device queue—See "NetCharDevQSetInfo" on page 3-263.
- Retrieving information about a serial device—See "NetCharDevGetInfo" on page 3-248.
- Retrieving information about a serial device queue—See "NetCharDevQGetInfo" on page 3-254.

NetCharDevEnum

The NetCharDevEnum (admin, server, DOS) function provides information on all available serial devices pooled in shared serial device queues on a server.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetCharDevEnum(servername, level, buf, buflen,
               entriesread, totalentries)
char far *           servername;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far * entriesread;
unsigned short far * totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (0 or 1) requested for the returned *chardev_info* data structure.
- *buf* points to the *chardev_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NoCommDevs	2337	There are no shared communication devices.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Related Information

For information on:

- Enum functions—See Chapter 1, "Overview of OS/2 LAN Server API."
- Listing serial device queues on a server—See "NetCharDevQEnum" on page 3-251.

NetCharDevGetInfo

The NetCharDevGetInfo (server, DOS) function retrieves information about a particular serial device in a shared serial device queue on a server.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetCharDevGetInfo(servername, devname, level, buf,
                  buflen, totalavail)
char far *           servername;
char far *           devname;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *devname* points to an ASCIIZ string containing the name of a serial device.
- *level* specifies the level of detail (0 or 1) requested for the returned *chardev_info* data structure.
- *buf* points to the *chardev_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_DevNotFound	2332	This device cannot be shared.
NERR_NoCommDevs	2337	There are no shared communication devices.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

A device can belong to more than one queue.

Related Information

For information on:

- GetInfo functions—See Chapter 1, "Overview of OS/2 LAN Server API."
- Listing all serial device queues—See "NetCharDevEnum" on page 3-246.
- Modifying the state of a serial device—See "NetCharDevControl" on page 3-243.

NetCharDevQEnum

The NetCharDevQEnum (server, DOS) function enumerates all serial device queues on a server.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetCharDevQEnum(servername, username, level, buf,
                buflen, entriesread, totalentries)
char far *            servername;
char far *            username;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  entriesread;
unsigned short far *  totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *username* points to an ASCIIZ string containing a user name. *username* can be used to calculate how many requests are pending ahead in the queue by examining the *cq1_numahead* parameter.
- *level* specifies the level of detail (0 or 1) requested for the *chardevQ_info* data structure.
- *buf* points to the returned *chardevQ_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NoCommDevs	2337	There are no shared communication devices.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFSRamClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Related Information

For information on:

- Deleting the contents of a serial device queue—See "NetCharDevQPurge" on page 3-257.
- Enum functions—See Chapter 1, "Overview of OS/2 LAN Server API."
- Listing serial devices on a server—See "NetCharDevEnum" on page 3-246.

NetCharDevQGetInfo

The NetCharDevQGetInfo (server, DOS) function retrieves information about a particular serial device queue on a server.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetCharDevQGetInfo(servername, queuename, username, level,
                   buf, buflen, totalavail)
char far *           servername;
char far *           queuename;
char far *           username;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *queuename* points to an ASCIIZ string containing the name of the serial device queue for which information is being requested.
- *username* points to an ASCIIZ string specifying the name of a user. *username* can be used to calculate how many requests are pending ahead in the queue by examining the *cq1_numahead* parameter.
- *level* is a short integer specifying the level of detail (0 or 1) requested for the *chardevQ_info* data structure.
- *buf* points to the *chardevQ_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.

Manifest	Value	Meaning
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_BadUsername	2202	The user name or group name parameter is invalid.
NERR_NoCommDevs	2337	The user does not belong to this group.
NERR_QueueNotFound	2338	A queue does not exist for this request.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Related Information

For information on:

- GetInfo functions—See Chapter 1, "Overview of OS/2 LAN Server API."
- Listing serial device queues on a server—See "NetCharDevQEnum" on page 3-251.
- Modifying the state of a serial device queue—See "NetCharDevQSetInfo" on page 3-263.

NetCharDevQPurge

The NetCharDevQPurge (admin, server, DOS) function deletes all pending requests on a serial device queue.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetCharDevQPurge(servername, queuename)
char far * servername;
char far * queuename;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *queuename* points to an ASCIIZ string containing the name of the queue to purge pending requests.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_SEM_TIMEOUT	121	A time out happened from the semaphore API functions.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.

Manifest	Value	Meaning
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_NoCommDevs	2337	There are no shared communication devices.
NERR_QueueNotFound	2338	A queue does not exist for this request.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- DosSemRequest.

Remarks

The NetCharDevQPurge function deletes only requests that have not yet been assigned to a device. A process that currently has a device open is unaffected. All pending requests queued on *queuename* are canceled, returning the ERROR_BAD_NET_RESP error code for each call to the DosOpen function. All handles are still valid.

Related Information

For information on:

- Closing the current session of a serial device—See "NetCharDevControl" on page 3-243.
- Deleting the contents of a serial device queue—See "NetCharDevQPurgeSelf" on page 3-260.
- Listing a serial device queues of a server—See "NetCharDevQEnum" on page 3-251.
- Modifying the state of a serial device queue—See "NetCharDevQSetInfo" on page 3-263.

NetCharDevQPurgeSelf

The NetCharDevQPurgeSelf (server, DOS) function deletes all pending requests waiting in a serial device queue that were submitted by a particular computer.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetCharDevQPurgeSelf(servername, queuename, computername)
char far * servername;
char far * queuename;
char far * computername;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *queuename* points to an ASCIIZ string containing the name of the queue to purge queue requests.
- *computername* points to an ASCIIZ string specifying the name of a computer whose requests are to be deleted from *queuename*.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_SEM_TIMEOUT	121	A time out happened from the semaphore API functions.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_ItemNotFound	2115	The device queue is empty.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadUsername	2202	The user name or group name parameter is invalid.
NERR_NoCommDevs	2337	There are no shared communication devices.
NERR_QueueNotFound	2338	A queue does not exist for this request.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- DosSemRequest.

Remarks

The NetCharDevQPurgeSelf function deletes all requests that the NetCharDevPurge function specifies, except that only requests from *computername* are deleted. A process that currently has a device open is unaffected.

Administrative permissions are required to delete requests from other computers when NetCharDevQPurgeSelf is called remotely. Administrative permissions are not required to delete requests made from the computer of an application.

Related Information

For information on:

- Listing serial device queues on a server—See "NetCharDevQEnum" on page 3-251.
- Modifying the state of a serial device queue—See "NetCharDevQSetInfo" on page 3-263.
- Deleting the contents of a serial device queue—See "NetCharDevQPurge" on page 3-257.

NetCharDevQSetInfo

The NetCharDevQSetInfo (admin, server, DOS) function modifies the state of a serial device queue on a server.

Syntax

```
#include <netcons.h>
#include <chardev.h>

unsigned far pascal
NetCharDevQSetInfo(servername, queuename, level,
                   buf, buflen, parmnum)
char far *      servername;
char far *      queuename;
short           level;
char far *      buf;
unsigned short  buflen;
short           parmnum;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *queuename* points to an ASCIIZ string containing the name of the serial device queue to set.
- *level* specifies the level of detail (1) supplied to the *chardevQ_info* data structure.
- *buf* points to the *chardevQ_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* determines whether *buf* contains a complete *chardevQ_info_1* data structure or a single data structure component. If *parmnum* is 0, then *buf* must contain a *chardevQ_info_1* data structure. Otherwise, *parmnum* must specify the ordinal position value for one of the following data structure components, as defined in CHARDEV.H as follows:

Manifest	Value	Component
CHARDEVQ_PRIORITY_PARMNUM	2	*cq1_priority*
CHARDEVQ_DEVICES_PARMNUM	3	*cq1_devs*

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.

Manifest	Value	Meaning
ERROR_SEM_TIMEOUT	121	A time out happened from the semaphore API functions.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_RedirectedPath	2117	The operation is invalid on a redirected device.
NERR_NoRoom	2119	The server is currently out of the requested resource.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.

Manifest	Value	Meaning
NERR_UseNotFound	2250	The connection cannot be found.
NERR_BadQueuePriority	2335	The queue priority is invalid.
NERR_NoCommDevs	2337	There are no shared communication devices.
NERR_QueueNotFound	2338	A queue does not exist for this request.
NERR_BadDevString	2340	This list of devices is invalid.
NERR_BadDev	2341	The requested device is invalid.
NERR_InUseBySpooler	2342	This device is already in use by the spooler.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosDevIOCtl
- DosFsRamSemClear
- DosFsRamSemRequest
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCrl(NIOCGETASGLIST)[-ERROR_NO_MORE_FILES]
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrg
- DosSemClear
- DosSemRequest
- redir.NIOCGETASGLIST1_1[-ERROR_NO_MORE_FILES].

Remarks

The NetCharDevSetInfo can set the *cq1_priority* and the *cq1_devs* components in the *chardevQ_info_1* data structure.

Related Information

For information on:

- Deleting the contents of a serial device queue—See "NetCharDevQPurge" on page 3-257.
- Listing serial device queues on a server—See "NetCharDevQEnum" on page 3-251.
- SetInfo functions—See Chapter 1, "Overview of OS/2 LAN Server API."

Server Category

NetServerAdminCommand (*admin, server, DOS*)—See "NetServerAdminCommand" on page 3-284.

NetServerDiskEnum (*admin, DOS*)—See "NetServerDiskEnum" on page 3-287.

NetServerEnum2 (*DOS*)—See "NetServerEnum2" on page 3-289.

NetServerGetInfo (*partially admin, server, DOS*)—See "NetServerGetInfo" on page 3-292.

NetServerSetInfo (*admin, server, DOS*)—See "NetServerSetInfo" on page 3-295.

With the functions in the Server category, an application can perform remote administrative tasks on a local or remote server. These functions are used with the SERVER.H and NETCONS.H include files.

DOS Considerations

Under DOS, the functions in the server category enable remote administrative tasks to be performed on a remote server. NetServerEnum can be executed on either a local requester or remote server; all of the other server functions are executed on a remote server. Attempting to execute NetServerAdminCommand or NetServerGetInfo on a local requester returns NERR_RemoteOnly.

Description

Any user or application assigned administrative privileges on a local or remote server can perform administrative tasks on that server, controlling its operation, user access, and resource sharing. A user can be given such privilege or access by way of the NetUserAdd and NetUserSetInfo functions (see "User Category" on page 3-382), or the NetShareAdd and NetShareSetInfo functions (see "Share Category" on page 3-337).

Certain low-level parameters affecting the operation of a server, defined in the IBMLAN.INI file of the server, can be examined and modified by calling the NetServerGetInfo and NetServerSetInfo functions.

Other changes in the operation of a server require execution of one of the NET commands (such as NetUse or NetShare). To execute a Net command on a server, an application calls the NetServerAdminCommand function. This function also accepts any OS/2 command for execution on the server.

To obtain a list of servers available to perform remote administration, an application calls the NetServerEnum2 function. The NetServerEnum2 function enumerates the set of all servers visible on the network. The type of NetServerEnum2 matches the bit mask in the field. To obtain a list of local drives, an application calls the NetServerDiskEnum function.

Data Structures

To set up a server or to reconfigure an existing server, use *server_info_1*, *server_info_2*, or *server_info_3* with NetServerSetInfo.

NetServerGetInfo returns configuration information at four levels by way of *server_info_0* (server name only), *server_info_1*, *server_info_2*. and *server_info_3*.

The lists of server information returned by NetServerEnum2 are limited to level 0 or level 1.

NetServerAdminCommand uses no data structure.

Server Information (Level 0)

```
struct server_info_0 {
   char sv0_name[CNLEN + 1];
};
```

where:

- *sv0_name* is an ASCIIZ string containing the name of a server.

Server Information (Level 1)

At level 1, NetServerEnum2, NetServerGetInfo, and NetServerSetInfo use the *server_info_1* data structure.

```
struct server_info_1 {
char          sv1_name[CNLEN + 1];
unsigned char sv1_version_major;
unsigned char sv1_version_minor;
unsigned long sv1_type;
char far *    sv1_comment;
};
```

where:

- *sv1_name* is an ASCIIZ string containing the name of a server.
- *sv1_version_major* is the major release version number of the OS/2 LAN Requester/Server software.
- *sv1_version_minor* is the minor release version number of the OS/2 LAN Requester/Server software.
- *sv1_type* indicates the type of software the computer is running, defined in SERVER.H as follows:

Manifest	Bit	Mask	Type of Software
SV_TYPE_WORKSTATION	1	0x00000001	Requester.
SV_TYPE_SERVER	2	0x00000002	Server.
SV_TYPE_DOMAIN_CTRL	4	0x00000008	Domain controller.
SV_TYPE_DOMAIN_BAKCTRL	5	0x00000010	Backup domain controller.
SV_TYPE_TIME_SOURCE	6	0x00000020	Time server.
SV_TYPE_ALL	All	0xFFFFFFFF	All types of servers.

Only user and group information is replicated to the backup domain controller.

- *sv1_comment*, which can be NULL, points to an ASCIIZ string containing a comment describing the server.

Server Information (Level 2)

```
struct server_info_2 {
   char           sv2_name[CNLEN+1];
   unsigned char  sv2_version_major;
   unsigned char  sv2_version_minor;
   unsigned long  sv2_type;
   char far *     sv2_comment;
   unsigned long  sv2_ulist_mtime;
   unsigned long  sv2_glist_mtime;
   unsigned long  sv2_alist_mtime;
   unsigned short sv2_users;
   unsigned short sv2_disc;
   char far *     sv2_alerts;
   unsigned short sv2_security;
   unsigned short sv2_auditing;
   unsigned short sv2_numadmin;
   unsigned short sv2_lanmask;
   unsigned short sv2_hidden;
   unsigned short sv2_announce;
   unsigned short sv2_anndelta;
   char           sv2_guestacct[UNLEN + 1];
   unsigned char  sv2_pad1;
   char far *     sv2_userpath;
   unsigned short sv2_chdevs;
   unsigned short sv2_chdevq;
   unsigned short sv2_chdevjobs;
   unsigned short sv2_connections;
   unsigned short sv2_shares;
   unsigned short sv2_openfiles;
   unsigned short sv2_sessopens;
   unsigned short sv2_sessvcs;
   unsigned short sv2_sessreqs;
   unsigned short sv2_opensearch;
   unsigned short sv2_activelocks;
   unsigned short sv2_numreqbuf;
   unsigned short sv2_sizreqbuf;
   unsigned short sv2_numbigbuf;
   unsigned short sv2_numfiletasks;
   unsigned short sv2_alertsched;
   unsigned short sv2_erroralert;
   unsigned short sv2_logonalert;
   unsigned short sv2_accessalert;
   unsigned short sv2_diskalert;
   unsigned short sv2_netioalert;
   unsigned short sv2_maxauditsz;
   char far *     sv2_srvheuristics;
};
```

where:

- *sv2_name* is an ASCIIZ string containing the name of a server.
- *sv2_version_major* is the major release version number of the OS/2 LAN Requester/Server software.

- *sv2_version_minor* is the minor release version number of the OS/2 LAN Requester/Server software.
- *sv2_type* indicates the type of software the computer is running, defined in SERVER.H as follows:

Manifest	Bit	Mask	Type of Software
SV_TYPE_WORKSTATION	1	0x00000001	Requester.
SV_TYPE_SERVER	2	0x00000002	Server.
SV_TYPE_DOMAIN_CTRL	4	0x00000008	Domain controller.
SV_TYPE_DOMAIN_BAKCTRL	5	0x00000010	Backup domain controller.
SV_TYPE_TIME_SOURCE	6	0x00000020	Time server.
SV_TYPE_ALL	All	0xFFFFFFFF	All types of servers.

- *sv2_comment*, which can be NULL, points to an ASCIIZ string containing a comment describing the server.
- *sv2_ulist_mtime* indicates the last time (in seconds from January 1,1970) the users list was modified.
- *sv2_glist_mtime* indicates the last time (in seconds from January 1, 1970) the groups list was modified.
- *sv2_alist_mtime* indicates the last time (in seconds from January 1, 1970) the access control list was modified.
- *sv2_users* indicates the number of users that are allowed on the server.
- *sv2_disc* indicates the auto-disconnect time (in minutes). A session is disconnected if it is idle longer than the time specified by *sv2_disc*. If *sv2_disc* is -1, auto-disconnect is not enabled.
- *sv2_alerts* points to an ASCIIZ string containing the list of user names on the alert table of the server. Spaces separate the names.
- *sv2_security* specifies the security type of the server. It is set to SV_USERSECURITY which is defined in SERVER.H.
- *sv2_auditing* indicates whether auditing is enabled on the server. If 0, auditing is disabled. If non-zero, the server is auditing all OS/2 LAN Requester/Server activities, as described in "Auditing Category" on page 3-43.
- *sv2_numadmin* indicates the number of administrators a server can accommodate.
- *sv2_lanmask* determines the order in which the network device drivers are served.

- *sv2_hidden* determines whether the server is visible to other computers in the same domain. *sv2_hidden* is defined in in SERVER.H as follows:

Manifest	Value	Meaning
SV_VISIBLE	0	Server.
SV_HIDDEN	1	Hidden server, not visible.

- *sv2_announce* specifies the network announce rate (in seconds), which determines how often the server will be announced to other computers on the network.
- *sv2_anndelta* specifies the random announce rate (in milliseconds) for *sv2_announce*. The announce interval (*sv2_announce*) can vary by the amount specified in *sv2_anndelta* (for example, it could vary from 9.99 seconds to 10.01 seconds instead of being exactly 10 seconds each time).
- *sv2_guestacct* is an ASCIIZ string containing the name of a server's reserved GUEST user account.
- *sv2_pad1* WORD-aligns the data structure components.
- *sv2_userpath* points to an ASCIIZ string containing the path name to user directories.
- *sv2_chdevs* indicates the number of serial devices that can be shared on the server.
- *sv2_chdevq* indicates the number of serial device queues that can coexist on the server.
- *sv2_chdevjobs* indicates the number of serial device jobs that can be pending on a server.
- *sv2_connections* indicates the number of connections to netnames that are allowed on a server.
- *sv2_shares* indicates the number of netnames a server can accommodate.
- *sv2_openfiles* indicates the number of files that can be opened at once.
- *sv2_sessopens* indicates the number of files that can be opened in one session.
- *sv2_sessvcs* indicates the maximum number of virtual circuits per client.
- *sv2_sessreqs* indicates the number of simultaneous requests that a client can make on any virtual circuit.
- *sv2_opensearch* indicates the number of searches that can be opened at once.
- *sv2_activelocks* indicates the number of file locks that can be active.
- *sv2_numreqbuf* indicates the number of server buffers that are provided.
- *sv2_sizreqbuf* indicates the size (in bytes) of each server buffer.
- *sv2_numbigbuf* indicates the number of 64KB server buffers that are provided.
- *sv2_numfiletasks* indicates the number of processes that can access the operating system at one time.
- *sv2_alertsched* indicates the alert interval (in seconds) for notifying an administrator of a network event.

- *sv2_erroralert* indicates the number of entries that can be written to the error log file during a *sv2_alertsched* interval before notifying an administrator.
- *sv2_logonalert* indicates the number of invalid logon attempts to allow a user before notifying an administrator.
- *sv2_accessalert* indicates the number of invalid file accesses to allow before issuing an administrative alert.
- *sv2_diskalert* indicates the point at which (the number of kilobytes of free disk space) an administrator must be notified that the free space of a disk is low.
- *sv2_netioalert* indicates the network I/O error ratio (in tenths of a percent) to allow before notifying an administrator.
- *sv2_maxauditsz* indicates the maximum audit file size (in kilobytes).
- *sv2_srvheuristics* points to an ASCIIZ string of flags used to control the operations of a server.

  The heuristics default to values that are optimal for most configurations and normally need not be changed. The default value for *sv2_srvheuristics* is defined in IBMLAN.INI as follows:

  ```
  srvheuristics=11110141111111011
  ```

  The maximum value for *sv2_srvheuristics* is defined in IBMLAN.INI as follows:

  ```
  srvheuristics = 12111191119119191
  ```

  If a partial string is specified, the default values are used for the remaining heuristics. If the string is NULL, or is not present in the IBMLAN.INI file, the default string is used.

  The result of using values other than those listed is undefined.

  The characters in the string (from left to right) have the following meaning. Unless otherwise defined, 0 turns off a heuristics feature, and 1 turns on the feature.

The parameters and their descriptions are as follows:

Parameter	Description
srvannounce	Specifies the rate, in seconds, at which the server announces its presence on the network. **Default value:** 60 **Minimum value:** 0 **Maximum value:** 65535
srvheuristics	Sets a variety of server fine-tuning options. Each digit has an independent meaning. Missing digits are assumed to be the defaults as described. Except where noted, each is a binary digit where 0 means *off* or *inactive*, while 1 means *on* or *active*. The following are the meanings of the digits:

Digit	Meaning
0	Use opportunistic locking when opening files. This lets the Server service assume that the first requester of the file is the only active process using that file. The server service will buffer the file and prevent a second requester from accessing the file until the buffer data is flushed. The default is 1. For opportunistic locking to occur, both this heuristic and **wrkheuristic** 0 in the requester must be active.
1	Use read-ahead (read additional data to try to guess what the requester may want) when the requester is performing sequential access, as follows: **Value** **Meaning** 0 Do not use read-ahead 1 Use single-thread read-ahead 2 Use asynchronous read-ahead thread. The default is 1. This heuristic pertains to reading ahead to the server's buffers (big buffers and requester buffers) from the file system and cache.
2	Use write-behind (tell the requester a write is completed before actually performing the write). If the write generates an error, the error appears on a subsequent write. Files opened for write-through will not use write-behind. The default is 1. This heuristic pertains to writing behind from the server's buffers (big buffers and requester buffers) to the file system and cache.
3	Use chain sends. The default is 1. For the chain send NETBIOS command to work, both this heuristic and **wrkheuristic** 8 in the requester must be active (set to 2, default).
4	Check all incoming server message blocks (SMBs) for correct format. This is useful with mixed versions and brands of network software on the LAN. The default is 0. To prevent wasted CPU cycles in an OS/2 LAN Server environment, leave this heuristic at the default.
5	Support file control block (FCB) opens (collapse all FCB opens for a file to a single open). This is only useful for DOS applications on the network. The default is 0.
6	Set the priority for the server. Lists the possible priority values (0 is the highest priority and 9 is the lowest).

Table 3-1. Server Priority		
Priority	**Class of Priority**	**Level of Class**
0	3	31
1	3	23
2	3	15
3	3	7
4	3	0
5	2	31
6	2	23
7	2	15
8	2	7
9	2	0

The default is 4.

Server priority is set to allow other applications to have CPU access also, if required.

7 Automatically allocate more memory for searches if it runs out, up to 2048 searches. If DOS requesters are on the network, set this to 1. The default is 1.

This heuristic pertains to directory searches (DosFindFirst). Memory is allocated dynamically instead of being locked up when it may not be needed.

8 Write records to the audit trail only when the scavenger wakes up. The scavenger is a high-priority server thread that monitors the network for errors, writes to the error log and audit trail, and sends alerts

When this is set to 0, any write to the audit trail wakes the scavenger. Heuristic 10 controls the wake-up interval of the scavenger. The default is 1.

9 Do full buffering (as controlled by **srvheuristics** 1 and 2) when a file is opened with deny-write sharing mode. When this is set to 0, deny write access has no buffering for any requester using this server. The default is 1.

If an application breaks using buffering of deny-write opened files, use this heuristic to disable buffering for all requesters.

10 Set the interval for the scavenger to wake up. The scavenger is a thread of the server process that performs the following tasks:

- Automatic disconnection of sessions

- Sending administrative alerts
- Writing to the audit trail file.

Set this entry as follows:

Value	Meaning
0	5 seconds
1	10 seconds
2	15 seconds
3	20 seconds
4	25 seconds
5	30 seconds
6	35 seconds
7	40 seconds
8	45 seconds
9	50 seconds.

The default is 1. Heuristic 8 can cause the scavenger to wake up at other times.

11 Allow compatibility-mode opens of certain types of files by translating them to sharing mode opens with deny-write. This is useful for sharing executable and other types of files. The default is 2.

This heuristic controls how strictly the server enforces compatibility opens for read only. In the strictest sense of compatibility opening, if there is any open of a file with a sharing mode set, or if another session has the file open in compatibility mode, a compatibility-mode open of that file fails.

The settings of this heuristic relax the strictness. The first level allows different Dos Requester machines to execute the same programs. The second level extends to batch files. The highest level translates compatibility-mode opens into a deny-none sharing mode open for read/write. Not all applications support this mode of operation.

Values for **srvheuristic** 11 include the following:

Value	Meaning
0	Do not use soft-compatibility opens.
1	Use a deny-none sharing mode on .EXE and .COM files.
2	Use a deny-none sharing mode on .EXE, .COM, .IMG and .BAT files.
3	Use a deny-none sharing mode on all compatibility-mode opens.

The default is 2.

12 Allow Dos Requester machines to use a second virtual circuit when doing printer requests. If this is not set, a second virtual circuit ends any previous sessions set up for that Dos Requester machine. The default is 1.

13 Set the number of 64KB buffers used for read-ahead. Possible values are 0 to 9, where 0 means read-ahead is disabled. If this is set to a value larger than **numbigbuf**, then it is reset to the value of **numbigbuf**−1.

Each 64KB buffer is divided into sixteen 4KB read-ahead buffers. You might want more than one big buffer allocated here if you are processing many files simultaneously with small reads. The default is 1.

Using 64KB (big buffers) for read-ahead involves a tradeoff between large file transfers and small-record read and write operations. Provided there are two 64KB buffers remaining in the server for each requester doing concurrent large file transfers, you can use the remaining 64KB buffers for read-ahead without a penalty.

14 Convert incoming path specifications into the most basic format that the OS/2 LAN Server understands. This conversion includes changing lowercase characters to uppercase, and slashes (used in path names) to backslashes (/ to \). The default is 0.

15 Set the time the server waits before breaking an opportunistic lock. You may want to set a longer time when the network is subject to long delays.

Table 3-2 shows possible values:

Table 3-2. Opportunistic Lock Timeout	
Value	**Time (seconds)**
0	35
1	70
2	140
3	210
4	280
5	350
6	420
7	490
8	560
9	640

The default is 0.

If a second requester requests opening of a locked file, the server notifies the first requester to flush buffers and prepare for unlocking. If the first requester does not respond within the time specified here, the server closes the first requester's open of the file.

The server can lock a file opened in deny-none sharing mode (as long as there are no other requests to access the file) so that buffering can be used to enhance performance. The server provides exclusive use of the file to the first requester, preventing the second requester from accessing the file until buffer data is flushed (written to disk). This heuristic defines when the server breaks the lock and grants access to the second requester.

16 Validate the input/output controls (IOCTLs) across the network. When this is set to 1 (on), the server accepts only generic device IOCTLs (categories 01H, 05H, and 0BH). See *IBM OS/2 Programming Tools and Information* for more information.

With this heuristic set to 0 (off), the server could receive invalid IOCTL pointers because of differences in device drivers between vendors. This can shut down the server. You may need to set this heuristic to 0 to use certain device drivers, such as custom-built drivers.

The default is 1.

17 Determines how long the server maintains unused dynamic big (64KB) buffers before freeing the memory. This digit can range from 0 through 9, with the following meanings:

Digit	Timeout
0	0 seconds (immediately after use)
1	1 second
2	10 seconds
3	1 minute
4	5 minutes
5	10 minutes
6	20 minutes
7	40 minutes
8	1 hour
9	Maintain big buffers indefinitely.

Default value: 1 (1 second).

18 Determines how long the server waits after failing to allocate a big (64KB) buffer before trying again. This digit can be from 0 to 5, with the following meanings:

Digit	Timeout
0	0 seconds (immediately after use)
1	1 second
2	10 seconds
3	1 minute
4	5 minutes
5	10 minutes.

Default value: 3 (1 minute).

srvnets

Lists names of the networks the server is to run on. Names of available networks are listed in the Networks section of the IBMLAN.INI file.

Required value: net1

srvpipes

Sets the maximum number of pipes that the server uses. Increase this number if many users log on simultaneously.

Default value: 3

Minimum value: 1

Maximum value: 20

srvservices

Specifies network services to start with the *server* service. This value is defined by the user at installation.

Server Information (Level 3)

```
struct server_info_3 {
   char           sv3_name[CNLEN+1];
   unsigned char  sv3_version_major;
   unsigned char  sv3_version_minor;
   unsigned long  sv3_type;
   char far *     sv3_comment;
   unsigned long  sv3_ulist_mtime;
   unsigned long  sv3_glist_mtime;
   unsigned long  sv3_alist_mtime;
   unsigned short sv3_users;
   unsigned short sv3_disc;
   char far *     sv3_alerts;
   unsigned short sv3_security;
   unsigned short sv3_auditing;
   unsigned short sv3_numadmin;
   unsigned short sv3_lanmask;
   unsigned short sv3_hidden;
   unsigned short sv3_announce;
   unsigned short sv3_anndelta;
   char           sv3_guestacct[UNLEN + 1];
   unsigned char  sv3_pad1;
   char far *     sv3_userpath;
   unsigned short sv3_chdevs;
   unsigned short sv3_chdevq;
   unsigned short sv3_chdevjobs;
   unsigned short sv3_connections;
   unsigned short sv3_shares;
   unsigned short sv3_openfiles;
   unsigned short sv3_sessopens;
   unsigned short sv3_sessvcs;
   unsigned short sv3_sessreqs;
   unsigned short sv3_opensearch;
   unsigned short sv3_activelocks;
   unsigned short sv3_numreqbuf;
   unsigned short sv3_sizreqbuf;
   unsigned short sv3_numbigbuf;
   unsigned short sv3_numfiletasks;
   unsigned short sv3_alertsched;
   unsigned short sv3_erroralert;
   unsigned short sv3_logonalert;
   unsigned short sv3_accessalert;
   unsigned short sv3_diskalert;
   unsigned short sv3_netioalert;
   unsigned short sv3_maxauditsz;
   char far *     sv3_srvheuristics;
   unsigned short sv3_auditedevents;
};
```

where:

- *sv3_name* is an ASCIIZ string containing the name of a server.
- *sv3_version_major* is the major release version number of the OS/2 LAN Requester/Server software.
- *sv3_version_minor* is the minor release version number of the OS/2 LAN Requester/Server software.

- *sv3_type* indicates the type of software the computer is running, defined in SERVER.H as follows:

Manifest	Bit	Mask	Type of Software
SV_TYPE_WORKSTATION	1	0x00000001	Requester.
SV_TYPE_SERVER	2	0x00000002	Server.
SV_TYPE_DOMAIN_CTRL	4	0x00000008	Domain controller.
SV_TYPE_DOMAIN_BAKCTRL	5	0x00000010	Backup domain controller.
SV_TYPE_TIME_SOURCE	6	0x00000020	Time server.
SV_TYPE_ALL	All	0xFFFFFFFF	All types of servers.

- *sv3_comment*, which can be NULL, points to an ASCIIZ string containing a comment describing the server.
- *sv3_ulist_mtime* indicates the last time (in seconds from January 1, 1970) the users list was modified.
- *sv3_glist_mtime* indicates the last time (in seconds from January 1, 1970) the groups list was modified.
- *sv3_alist_mtime* indicates the last time (in seconds from January 1, 1970) the access control list was modified.
- *sv3_users* indicates the number of users allowed on the server.
- *sv3_disc* indicates the auto-disconnect time (in minutes). A session is disconnected if it is idle longer than the time specified by *sv3_disc*. If *sv3_disc* is -1, auto-disconnect is not enabled.
- *sv3_alerts* points to an ASCIIZ string containing the list of user names on the alert table of the server. Spaces separate the names.
- *sv3_security* specifies the security type of the server; the value is set to SV_USERSECURITY.
- *sv3_auditing* indicates whether auditing is enabled on the server. If 0, auditing is disabled. If non-zero, the server is auditing all OS/2 LAN Requester/Server activities, as described in "Auditing Category" on page 3-43.
- *sv3_numadmin* indicates the number of administrators that a server can accommodate.
- *sv3_lanmask* determines the order in which the network device drivers are served.
- *sv3_hidden* is set to SV_VISIBLE which is defined in SERVER.H.
- *sv3_announce* specifies the network announce rate (in seconds), which determines how often the server will be announced to other computers on the network.
- *sv3_anndelta* specifies the random announce rate (in milliseconds) for *sv3_announce*. The announce interval (*sv3_announce*) can vary by the amount

specified in *sv3_anndelta* (for example, it could vary from 9.99 seconds to 10.01 seconds instead of being exactly 10 seconds each time).

- *sv3_guestacct* is an ASCIIZ string containing the name of the reserved GUEST user account of a server.
- *sv3_pad1* WORD-aligns the data structure components.
- *sv3_userpath* points to an ASCIIZ string containing the path name to user directories.
- *sv3_chdevs* indicates the number of serial devices that can be shared on the server.
- *sv3_chdevq* indicates the number of serial device queues that can coexist on the server.
- *sv3_chdevjobs* indicates the number of serial device jobs that can be pending on the server.
- *sv3_connections* indicates the number of connections to netnames that are allowed on a server.
- *sv3_shares* indicates the number of netnames that a server can accommodate.
- *sv3_openfiles* indicates the number of files that can be open at once.
- *sv3_sessopens* indicates the number of files that can be open in one session.
- *sv3_sessvcs* indicates the maximum number of virtual circuits per client.
- *sv3_sessreqs* indicates the number of simultaneous requests that a client can make on any virtual circuit.
- *sv3_opensearch* indicates the number of searches that can be opened at once.
- *sv3_activelocks* indicates the number of file locks that can be active.
- *sv3_numreqbuf* indicates the number of server buffers provided.
- *sv3_sizreqbuf* indicates the size (in bytes) of each server buffer.
- *sv3_numbigbuf* indicates the number of 64KB server buffers provided.
- *sv3_numfiletasks* indicates the number of processes that can access the operating system at one time.
- *sv3_alertsched* indicates the alert interval (in seconds) for notifying an administrator of a network event.
- *sv3_erroralert* indicates the number of entries that can be written to the error log file during a *sv3_alertsched* interval before notifying an administrator.
- *sv3_logonalert* indicates the number of invalid logon attempts to allow a user before notifying an administrator.
- *sv3_accessalert* indicates the number of invalid file accesses to allow before issuing an administrative alert.
- *sv3_diskalert* indicates the point (number of kilobytes of free disk space) at which an administrator must be notified that the free space of a disk is low.
- *sv3_netioalert* indicates the network I/O error ratio (in tenths of a percent) to allow before notifying an administrator.
- *sv3_maxauditsz* indicates the maximum audit file size (in kilobytes).

- *sv3_srvheuristics* points to an ASCIIZ string of flags used to control the operations of a server. See the heuristics information under "Server Information (Level 2)."
- *sv3_auditedevents* is the audit event control mask as follows:

Manifest	Meaning
SVAUD_SERVICE	Service state change.
SVAUD_GOODSESSLOGON	Successful session logon requests.
SVAUD_BADSESSLOGON	Unsuccessful session logon requests.
SVAUD_SESSLOGON	All session logon and logoff requests.
SVAUD_GOODNETLOGON	Successful network logon requests.
SVAUD_BADNETLOGON	Unsuccessful network logon requests.
SVAUD_NETLOGON	All network logon and logoff requests.
SVAUD_LOGON	All logon and logoff requests (network and session).
SVAUD_GOODUSE	Successful share requests.
SVAUD_BADUSE	Unsuccessful share requests.
SVAUD_USE	All share requests, regardless of gooduse or baduse switches.
SVAUD_USERLIST	Changes to the user or group account database.
SVAUD_PERMISSIONS	Changes to the access control list database.
SVAUD_RESOURCE	Resource access as defined by the per-resource auditing options specified in the access control list.
SVAUD_LOGONLIM	Logon limit violations.

Level 3 is not valid for the NetServerEnum2 call.

The new field *sv3_auditedevents* is settable.

For details on *sv3_auditedevents*, see *ae_type*, audit type, in "Auditing Category" on page 3-43.

Related Information

For information on:

- Domains—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- Remote administration of Net commands—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- Server heuristics—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*

NetServerAdminCommand

The NetServerAdminCommand (admin, server, DOS) function executes a command on a server.

Syntax

```
#include <netcons.h>
#include <server.h>

unsigned far pascal
NetServerAdminCommand(servername, command, result, buf, buflen,
                      bytesread, totalavail)
char far *            servername;
char far *            command;
short far *           result;
char far *            buf;
unsigned short        buflen;
unsigned short far *  bytesread;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *command* points to an ASCIIZ string containing the command to execute.
- *result* points to the returned exit code of the executed command.
- *buf* points to the output of the returned command.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *bytesread* points to an unsigned short integer indicating the number of bytes of information that were returned to *buf*.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
ERROR_FILENAME_EXCED_RANGE	206	The file name is longer than 8 characters or the extension is longer than 3 characters.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_NoRoom	2119	The server is currently out of the requested resource.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_TmpFile	2316	A failure occurred when opening a remote temporary file.

Manifest	Value	Meaning
NERR_TooMuchData	2317	The data returned from a remote administration command has been truncated to 64KB.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CantType	2357	The type of input cannot be determined.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- redir.GetNetInitPath.

Remarks

The NetServerAdminCommand function is a remote form of the C language library *system*() function.

When executed remotely, NetServerAdminCommand sets the environment of the server as follows:

- The default drive and directory is c:\IBMLAN\NETPROG
- The PATH environment is not set (NULL).

If executed locally, NetServerAdminCommand sets the environment of the server as follows:

- The current drive and directory of the caller are used
- The PATH environment is set to the user's default path.

Related Information

For information on:

- Listing available servers—See "NetServerEnum2" on page 3-289.
- Executing a program on a remote server—See "NetRemoteExec" on page 3-200.

NetServerDiskEnum

The NetServerDiskEnum (admin, DOS) function retrieves a list of disk drives on a workstation.

Syntax

```
#include <netcons.h>
#include <server.h>

unsigned far pascal
NetServerDiskEnum (servername, level, buf,
                   buflen, entriesread, totalentries)
char far *           servername;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far * entriesread;
unsigned short far * totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local workstation.
- *level* is a short integer that must be zero.
- *buf* points to the returned list of disk drive names.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_InternalError	2140	An internal error has occurred.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Remarks

The NetServerDiskEnum function returns a list of local drive names for the specified workstation. The drive names in the list are consecutive strings, each containing a drive letter, a colon (:), and a NULL string terminator (\0). For example, the following can be returned for a server having two floppy drives (A: and B:), one hard drive (C:), and one RAM drive (E:):

```
A:\0B:\0C:\0E:\0
```

Related Information

For information on:

- Listing the shared resources of a server—See "NetServerEnum2" on page 3-289.
- Listing available servers—See "NetServerEnum2" on page 3-289.

NetServerEnum2

The NetServerEnum2 (DOS) function enumerates the set of all servers visible on the network. The type of NetServerEnum2 matches the bit mask in the field.

Syntax

```
#include <netcons.h>
#include <server.h>

unsigned far pascal
NetServerEnum2(servername, level, buf, buflen,
               entriesread, totalentries, type, domain)
char far *            servername;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  entriesread;
unsigned short far *  totalentries;
unsigned long         type
char far *            domain;
```

where:

- *servername* points to an ASCIIZ string containing the name of a remote computer on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* is a short integer that specifies the level of detail (0 or 1) for the *server_info* data structure.

  Levels 2 and 3 are not valid for NetServerEnum2.
- *buf* points to the returned *server_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.
- *type* is the bit mask of types to find; defined as follows:

Manifest	Bit Mask
SV_TYPE_WORKSTATION	0x00000001
SV_TYPE_SERVER	0x00000002
SV_TYPE_DOMAIN_CTRL	0x00000008
SV_TYPE_DOMAIN_BAKCTRL	0x00000010
SV_TYPE_TIME_SOURCE	0x00000020
SV_TYPE_ALL	0xFFFFFFFF

- *domain* points to the servers in this domain.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_BrowserTableIncomplete	2319	The server table was initialized incorrectly.
NERR_NotLocalDomain	2320	This domain is not active on this computer.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosSemClear.

Remarks

This function can obtain only level 0 and 1 data structures. The *type* parameter is tested (nondestructive bitwise AND) against the *svx_type* field of each entry. Only entries that match at least one of the specified bits are included in the entries returned in the buffer and in the counts placed in *entriesread* and *totalentries*.

This API returns servers that are in the domain named in the domain argument. The named domain must be the domain of the requester, one of the *oth_domains* of the requester, or the *logon_domains* of the requester. If *domain* is NULL, the servers in all domains listed above are returned. If the domain is not one of the above values, the error NERR_NotLocalDomain is returned. If the domain argument is not NULL, *entriesread* and *totalentries* reflect the entries in the named domain.

The restrictions on the value of domain apply to the computer on which the NetServerEnum2 API is actually executed, that is, the local computer if the *servername* argument is NULL, or the named server if the *servername* argument is not NULL.

NetServerGetInfo

The NetServerGetInfo (partially admin, server, DOS) function retrieves information about a particular server.

Syntax

```
#include <netcons.h>
#include <server.h>

unsigned far pascal
NetServergetinfo(servername, level, buf, buflen, totalavail)
char far *          servername;
short               level;
char far *          buf;
unsigned short      buflen;
unsigned short far * totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (0, 1, 2, or 3) for the *server_info* data structure.
- *buf* points to the returned *server_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

Depending on the level of information requested (by way of the *level* parameter), the NetServerGetInfo function returns information ranging from the name of the server to a description of the server heuristics, which control the manner in which the server operates.

Level 0 and 1 information can be accessed remotely by ordinary users.

Related Information

For information on:

- Configuring a server—See "NetServerSetInfo" on page 3-295.
- Server heuristics—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*

NetServerSetInfo

The NetServerSetInfo (admin, server, DOS) function sets operating parameters for a server (individually or collectively).

Syntax

```
#include <netcons.h>
#include <server.h>

unsigned far pascal
NetServerSetInfo(servername, level, buf, buflen, parmnum)
char far *      servername;
short           level;
char far *      buf;
unsigned short  buflen;
short           parmnum;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (1, 2 or 3) to be provided in the *server_info* data structure.
- *buf* points to the data structure if *parmnum* is zero. Otherwise, *buf* points to the specific data component that will be changed.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* determines whether buf contains a complete *server_info* data structure or a single data structure component. If *parmnum* is 0, *buf* must contain a *server_info_1*, *server_info_2*, or *server_info_3* data structure. Otherwise, *parmnum* must specify the ordinal position value for one of the following data structure components, defined in SERVER.H as follows:

Manifest	Value	Component
SV_COMMENT_PARMNUM	5	*sv1_comment* or *sv2_comment*
SV_DISC_PARMNUM	10	*sv2_disc*
SV_ALERTS_PARMNUM	11	*sv2_alerts*
SV_HIDDEN_PARMNUM	16	*sv2_hidden*
SV_ANNOUNCE_PARMNUM	17	*sv2_announce*
SV_ANNDELTA_PARMNUM	18	*sv2_anndelta*
SV_ALERTSCHED_PARMNUM	37	*sv2_alertsched*
SV_ERRORALERT_PARMNUM	38	*sv2_erroralert*
SV_LOGONALERT_PARMNUM	39	*sv2_logonalert*
SV_ACCESSALERT_PARMNUM	40	*sv2_accessalert*
SV_DISKALERT_PARMNUM	41	*sv2_diskalert*
SV_NETIOALERT_PARMNUM	42	*sv2_netioalert*

Manifest	Value	Component
SV_MAXAUDITSZ_PARMNUM	43	*sv2_maxauditsz*

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_NoRoom	2119	The server is currently out of the requested resource.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.

Manifest	Value	Meaning
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_NetNameNotFound	2310	This shared resource does not exist.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Related Information

For information on retrieving the configuration of a server, see "NetServerGetInfo" on page 3-292.

Service Category

NetServiceControl (*partially admin, DOS*)—See "NetServiceControl" on page 3-310.

NetServiceEnum (*DOS*)—See "NetServiceControl" on page 3-310.

NetServiceGetInfo—See "NetServiceGetInfo" on page 3-317.

NetServiceInstall (*admin, DOS*)—See "NetServiceInstall" on page 3-320.

NetServiceStatus—See "NetServiceStatus" on page 3-323.

The functions in the Service category start and control network service programs. All of these functions can be called on local machine with ordinary user's privilege. For remote execution, administrative privilege is required except for the NetServiceEnum function. The functions in this category are used with the SERVICE.H and NETCONS.H include files.

Description

A service is a program of any size and function that other applications can use to perform some set of tasks on the network. An application starts and controls the operation of services with the functions in the Service category.

The OS/2 LAN Requester/Server software provides ten standard services. The two most important of these are *requester* and *server*, which provide the majority of the software required to operate a local area network. When booting the OS/2 LAN Requester/Server software, the *requester* and *server* services are started first, then the services defined in the *service* section of the IBMLAN.INI file are started, followed by the *server* service. The *netlogon* service on the servers (primary domain controller, backup domain controller, or member of a domain) are started after the *server* service has been started.

The following are descriptions of the tasks performed by the network services:

Service	Description
alerter	This service provides a system for notifying registered clients of certain classes of defined system events. The alerter service registers itself to receive all *print*, *errorlog*, and *admin* alerts, creating and registering a mailslot to receive alerts triggered by these events. When the *alerter* service receives an alert, it converts the information to text and sends a message to all clients registered for that class of event by calling NetMessageBufferSend. (Print alerts regarding an individual print job are sent only to the user who submitted the print job.) For *admin* or *errorlog* alerts, the *alerter* service sends a message to all users listed in the *sv2_alerts* component of the *server_info_2 data* structure. (A list of these users can be retrieved by calling the NetServerGetInfo function.) For more information on alert functions, see "Alert Category" on page 3-29.

Service	Description
requester	This service is a primary one. It maintains most internal information and activates the network device drivers. If the *requester* service is paused, no redirection can be established to printers or serial devices. All currently redirected and opened devices can be used and closed, but not reopened, during a *requester* pause. If an application calls DosOpen while the *requester* service is paused, the function opens a local device of the specified name instead of the intended redirected device. The *requester* service cannot be removed while the *server* service is started.
dlrinst	This service downloads the DOS LAN Requester code from the server to the DOS LAN requesters.
messenger	This service enables the receiving and logging of messages sent among requesters by calling the Message category functions (see "Message Category" on page 3-157). The *messenger* service receives messages sent from remote computers by way of NetMessageBufferSend or NetMessageFileSend functions. For more details, see the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
netlogon	This service provides logon verification. The *server* service must be started before *netlogon* can be started.
netpopup	This service displays messages on-screen.
netrun	This service supports the remote execution of processes. If the *netrun* service is paused, requests to remotely execute a program are denied. When the service is removed, each process that the service started is ended by the OS/2 DosKillProc function. The *server* service must be started before the *netrun* service can be started.
pcdosrpl	This service enables the server service to support remote initial program load (remote IPL) of DOS requesters.
replicator	This service provides a file replication service.
server	This service provides the basic functionality needed to share local resources over the network. If the *server* service is paused, all further requests for resources are denied. However, all current uses of shared resources continue.

Time Hints for Starting and Stopping a Service

For the services which take a long time to start or to stop, a mechanism is provided to feedback or "hint" to the process that started or stopped them. This mechanism allows the process to dynamically determine how long to wait for the service to complete the startup or stop, and in fact to provide the estimated time to the user, if determined by the application. The service can set the text field *svci2_text* as part of its status information.

The services and programs that start or stop the services can use this mechanism to communicate during the start and stop operations. The control completion pending (CCP) code values are used when a service expects to take a long time to start or stop. Any service that has a nontrial initialization or shutdown task can use this method.

If a service is not using CCP codes, the *code* field should always be set to zero while the service status is INSTALL_PENDING or UNINSTALL_PENDING.

The CCP codes divide the *code* field into fields as follows:

```
33322222 222221111 111111
21098765 432109876 54321098 76543210

xxxxxxxx xxxxxxxxH TTTTTTTT CCCCCCCC


x = not used (must be zero)
H = hint is given            SERVICE_CCP_QUERY_HINT
T = time to wait             SERVICE_CCP_WAIT_TIME
C = check point number       SERVICE_CCP_CHKPT_NUM
```

Bits that are not used must be set to zero.

Time to wait is the expected time to complete the current operation (start or stop), in tenths of a second.

Check point number is a number that should be incremented, or at least changed to a higher value, each time the service calls NetServiceStatus. A service should call NetServiceStatus fairly often to keep updating this number, remembering that it is an 8-bit quantity.

A controlling application that notes this check point value constantly changing will assume the service is still active. For that reason, the code that calls NetServiceStatus to update the CCP code should be in the main code path, and not in some time-triggered thread that might continue even while the current operation has halted due to an error.

The "Hint" bit informs the application controlling the service that the other (time and count) information is valid. If this bit is set, the field *svc2_text* can contain an ASCIIZ string that, when displayed to a user, provides some information about the current state of the service. The service can continue to use a NULL string if it wishes to provide no text. It is up to the controlling application to make use of the text field, but it can assume that the text is suitable for display to the user.

DOS Considerations

Under DOS, the functions are executed on a local requester. Attempting to execute the functions on a remote server returns ERROR_NOT_SUPPORTED.

Under DOS, the services cannot be completely stopped; however, they can be paused and continued using NetServiceControl.

Data Structures

The functions in the Service category use two types of data structures, one type containing basic status information, and the other containing different levels of detail. The NetServiceControl and NetServiceInstall functions use the *service_info_2* data structure. The NetServiceEnum and NetServiceGetInfo functions use the *service_info* data structures (level 0, 1 or 2). The NetServiceEnum returns information at three levels of detail (0 1, or 2); the *level* parameter controls the level of information returned. The NetServiceStatus function uses the *service_status* data structure.

Service Status

The NetServiceStatus function uses the following data structure:

```
struct service_status {
   unsigned short svcs_status;
   unsigned long  svcs_code;
   unsigned short svcs_pid;
   char           svcs_text[STXTLEN+1];
};
```

where:

- *svcs_status* specifies the status of the service. See *svci1_status* under Service Information (Level 1) for the possible values of this field.
- *svcs_code* is the error code returned if the designated service stops or fails to start properly.
- *svcs_pid* is the program identification number (PID) of a service.
- *svcs_text* is a NULL reserved ASCIIZ string, unless the service specified by *svcs_pid* is stopped. In this case, *svcs_text* must specify a parameter string related to the *svcs_code* component.

Service Information (Level 0)

```
struct service_info_0 {
   char svci0_name[SNLEN+1];
};
```

where *svci0_name* is an ASCIIZ string containing the name of the network service to monitor.

Service Information (Level 1)

```
struct service_info_1 {
char          svci1_name[SNLEN+1];
unsigned short svci1_status;
unsigned long  svci1_code;
unsigned short svci1_pid;
};
```

where:

- *svci1_name* is an ASCIIZ string containing the name of the network service to monitor.
- *svci1_status* is a bit map that indicates the status of the network service. The bits of *svci1_status* are defined as follows:

Manifest/Bit	Bit Mask	Meaning
SERVICE_INSTALL_STATE	0x03	The service is currently in one of the following start states:
Start State	**Value**	**Meaning**
SERVICE_UNINSTALLED	0x00	Service stopped.
SERVICE_INSTALL_PENDING	0x01	Service start pending.
SERVICE_UNINSTALL_PENDING	0x02	Service stop pending.
SERVICE_INSTALLED	0x03	Service started.
SERVICE_PAUSE_STATE	0x0C	The service is currently in one of the following pause states:
Pause State	**Value**	**Meaning**
SERVICE_ACTIVE	0x00	Service active.
SERVICE_CONTINUE_PENDING	0x04	Service continue pending.
SERVICE_PAUSE_PENDING	0x08	Service pause pending.
SERVICE_PAUSED	0x0C	Service paused.
Bit 4		Indicates whether the service can be removed by an application, as indicated by one of the following bit settings:
Bit Setting	**Value**	**Meaning**

Manifest/Bit	Bit Mask	Meaning
SERVICE_NOT_UNINSTALLABLE	0x00	Service cannot be removed.
SERVICE_UNINSTALLABLE	0x10	Service can be removed.
Bit 5		Indicates whether the service can be paused by an application, as indicated by one of the following bit settings:
Bit Setting	**Value**	**Meaning**
SERVICE_PAUSABLE	0x20	Service can be paused.
SERVICE_NOT_PAUSABLE	0x00	Service cannot be paused.
Bit 6-7		Reserved, with a value of 0.
Bit 8-10		Indicates whether particular tasks within the *requester* service have been paused, as indicated by one of the following bit settings:
Bit Setting	**Value**	**Meaning**
SERVICE_REDIR_PAUSED	0x700	Redirector paused.
SERVICE_REDIR_DISK_PAUSED	0x100	Redirector for disks paused.
SERVICE_REDIR_PRINT_PAUSED	0x200	Redirector for spooled devices paused.
SERVICE_REDIR_COMM_PAUSED	0x400	Redirector for serial devices paused.
Bit 11-15		Reserved.

- *svcil_code* specifies an error code when a service stops or fails to start properly.

  For stopped services (SERVICE_UNINSTALL), the high word of *svcil_code* defines primary error codes and the low word of *svcil_code* defines secondary error codes. High-word values of *svcil_code* are defined as follows:

Primary Error Code	Value	Meaning
SERVICE_UIC_NORMAL	0	Normal.

Primary Error Code	Value	Meaning
SERVICE_UIC_BADPARMVAL	3051	Incorrect parameter value specified.
SERVICE_UIC_MISSPARM	3052	Missing parameter.
SERVICE_UIC_UNKPARM	3053	Unknown parameter specified.
SERVICE_UIC_RESOURCE	3054	Insufficient resource.
SERVICE_UIC_CONFIG	3055	Configuration faulty.
SERVICE_UIC_SYSTEM	3056	OS/2 program error.
SERVICE_UIC_INTERNAL	3057	Internal error encountered.
SERVICE_UIC_AMBIGPARM	3058	Ambiguous parameter name.
SERVICE_UIC_DUPPARM	3059	Parameter duplicated.
SERVICE_UIC_KILL	3060	Ended by NetServiceControl when it did not respond.
SERVICE_UIC_EXEC	3061	Could not execute service program file.
SERVICE_UIC_SUBSERV	3062	Subservice failed to start.
SERVICE_UIC_CONFLPARM	3063	Conflict in the value or use of these parameters.
SERVICE_UIC_BADCOMPNAME	3064	Not a valid computer name.

Low-word values of *svcil_code* are defined as follows:

Secondary Error Code	Value	Meaning
SERVICE_UIC_M_NULL	0	Normal.
SERVICE_UIC_M_MEMORY	3070	Insufficient memory.
SERVICE_UIC_M_DISK	3071	Insufficient disk space.
SERVICE_UIC_M_THREADS	3072	Unable to create thread.
SERVICE_UIC_M_PROCESSES	3073	Unable to create process.
SERVICE_UIC_M_SECURITY	3074	Security failure.
SERVICE_UIC_M_LANROOT	3075	Incorrect or missing default path.
SERVICE_UIC_M_REDIR	3076	Network software not started.
SERVICE_UIC_M_SERVER	3077	Server software not started.
SERVICE_UIC_M_SEC_FILE_ERR	3078	Server could not access UAS database.
SERVICE_UIC_M_FILES	3079	Not supported.
SERVICE_UIC_M_LOGS	3080	Invalid IBMLAN\LOGS directory.

Secondary Error Code	Value	Meaning
SERVICE_UIC_M_LANGROUP	3081	Domain specified could not be used.
SERVICE_UIC_M_MSGNAME	3082	Computer name being used as a message name on another computer.
SERVICE_UIC_M_ANNOUNCE	3083	Requester failed to announce the server name.
SERVICE_UIC_M_UAS	3084	The UAS database is not configured correctly.

For start or stop pending (SERVICE_INSTALL_PENDING, SERVICE_UNINSTALL_PENDING) services, the bits of *svci1_code* are defined in SERVICE.H as follows:

Start Pending Code	Bit Mask	Meaning
SERVICE_CCP_NO_HINT	0x0	No reason given for start pending.
SERVICE_CCP_CHKPT_NUM	0xFF	Checkpoint number incremented each time the service calls the NetServiceStatus function (installer assumes incrementing denotes a valid service).
SERVICE_CCP_WAIT_TIME	0xFF00	Time to wait: expected time (tenths of a second) to start.
SERVICE_CCP_QUERY_HINT	0x10000	Reason given for start pending.

- *svci1_pid* is a program identification number (PID) for a service.

Service Information (Level 2)

```
struct service_info_2 {
char           svci2_name[SNLEN+1];
unsigned short  svci2_status;
unsigned long   svci2_code;
unsigned short  svci2_pid;
char           svci2_text[STXTLEN+1];
};
```

where:

- *svci2_name* is an ASCIIZ string containing the name of the network service to monitor.
- *svci2_status* specifies the status of *svci2_name*. The bits of *svci2_status* are defined as follows:

Manifest/Bit	Bit Mask	Meaning
SERVICE_INSTALL_STATE	0x03	The service is currently in one of the following start states:
Start State	**Value**	**Meaning**
SERVICE_UNINSTALLED	0x00	Service stopped.
SERVICE_INSTALL_PENDING	0x01	Service start pending.
SERVICE_UNINSTALL_PENDING	0x02	Service stop pending.
SERVICE_INSTALLED	0x03	Service started.
SERVICE_PAUSE_STATE	0x0C	The service is currently in one of the following pause states:
Pause State	**Value**	**Meaning**
SERVICE_ACTIVE	0x00	Service active.
SERVICE_CONTINUE_PENDING	0x04	Service continue pending.
SERVICE_PAUSE_PENDING	0x08	Service pause pending.
SERVICE_PAUSED	0x0C	Service paused.
4		Indicates whether the service can be removed by an application, as indicated by one of the following bit settings:
Bit Setting	**Value**	**Meaning**
SERVICE_NOT_UNINSTALLABLE	0x00	Service cannot be removed.
SERVICE_UNINSTALLABLE	0x10	Service can be removed.
5		Indicates whether the service can be paused by an application, as indicated by one of the following bit settings:
Bit Setting	**Value**	**Meaning**
SERVICE_PAUSABLE	0x20	Service can be paused.

Manifest/Bit	Bit Mask	Meaning
SERVICE_NOT_PAUSABLE	0x00	Service cannot be paused.
6-7		Reserved, with a value of 0.
8-10		Indicates whether particular tasks within the *requester* service have been paused, as indicated by one of the following bit settings:

Bit Setting	Value	Meaning
SERVICE_REDIR_PAUSED	0x700	Redirector paused.
SERVICE_REDIR_DISK_PAUSED	0x100	Redirector for disks paused.
SERVICE_REDIR_PRINT_PAUSED	0x200	Redirector for spooled devices paused.
SERVICE_REDIR_COMM_PAUSED	0x400	Redirector for serial devices paused.

11-15		Reserved.

- *svci2_code* specifies an error code when a service stops or fails to start properly. For stopped services (SERVICE_UNINSTALL), the high word of *svci2_code* defines primary error codes and the low word of *svci2_code* defines secondary error codes. The high word values of *svci2_code* are defined as follows:

Primary Error Code	Value	Meaning
SERVICE_UIC_NORMAL	0	Normal.
SERVICE_UIC_BADPARMVAL	3051	Incorrect parameter value specified.
SERVICE_UIC_MISSPARM	3052	Missing parameter.
SERVICE_UIC_UNKPARM	3053	Unknown parameter specified
SERVICE_UIC_AMBIGPARM	3058	Ambiguous parameter name.
SERVICE_UIC_DUPPARM	3059	Duplicated parameter.
SERVICE_UIC_RESOURCE	3054	Insufficient resource.
SERVICE_UIC_CONFIG	3055	Configuration faulty.
SERVICE_UIC_SYSTEM	3056	OS/2 program error.
SERVICE_UIC_INTERNAL	3057	Internal error encountered.

Primary Error Code	Value	Meaning
SERVICE_UIC_KILL	3060	Ended by the NetServiceControl function.
SERVICE_UIC_EXEC	3061	Could not execute service program file.
SERVICE_UIC_SUBSERV	3062	Subservice did not start.
SERVICE_UIC_CONFLPARM	3063	Conflict in the value or use of these parameters.
SERVICE_UIC_BADCOMPNAME	3064	Not a valid computer name.

The low-word values of *svci2_code* are defined as follows:

Secondary Error Code	Value	Meaning
SERVICE_UIC_M_NULL	0	Normal.
SERVICE_UIC_M_MEMORY	3070	Insufficient memory.
SERVICE_UIC_M_DISK	3071	Insufficient disk space.
SERVICE_UIC_M_THREADS	3072	Unable to create thread.
SERVICE_UIC_M_PROCESSES	3073	Unable to create process.
SERVICE_UIC_M_SECURITY	3074	Security failure.
SERVICE_UIC_M_LANROOT	3075	Incorrect or missing default path.
SERVICE_UIC_M_REDIR	3076	Network software not started.
SERVICE_UIC_M_SERVER	3077	Server software not started.
SERVICE_UIC_M_SEC_FILE_ERR	3078	Server could not access UAS database.
SERVICE_UIC_M_FILES	3079	Not supported.
SERVICE_UIC_M_LOGS	3080	Invalid IBMLAN\LOGS directory.
SERVICE_UIC_M_LANGROUP	3081	Domain specified could not be used.
SERVICE_UIC_M_MSGNAME	3082	Computer name being used as a message name on another computer.
SERVICE_UIC_M_ANNOUNCE	3083	Requester did not announce the server name.
SERVICE_UIC_M_UAS	3084	The UAS database is not configured correctly.

For start pending (SERVICE_INSTALL_PENDING/SERVICE_UNINSTALL_PENDING) services, the bits of *svci2_code* are defined in SERVICE.H as follows:

Start Pending Code	Bit Mask	Meaning
SERVICE_CCP_NO_HINT	0x0	No reason given for start pending.
SERVICE_CCP_CHKPT_NUM	0xFF	Checkpoint number incremented each time the service calls the NetServiceStatus function (installer assumes incrementing denotes a valid service).
SERVICE_CCP_WAIT_TIME	0xFF00	Time to wait: expected time (tenths of a second) to start.
SERVICE_CCP_QUERY_HINT	0x10000	Reason given for start or stop pending.

- *svci2_pid* specifies the program identification number of a service.
- *svci2_text* is a NULL reserved ASCIIZ string, except for stopped services (SERVICE_UNINSTALLED). In this case, *svci2_text* specifies a related parameter string for the *svci2_code* component. *svci2_text* cannot be longer than STXTLEN + 1 bytes in length.

NetServiceControl

The NetServiceControl (admin, DOS) function controls the operations of network services.

Syntax

```
#include <netcons.h>
#include <service.h>

unsigned far pascal
NetServiceControl(servername, service, opcode,
                  arg, buf, buflen)
char far *      servername;
char far *      service;
unsigned char   opcode;
unsigned char   arg;
char far *      buf;
unsigned short  buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *service* points to an ASCIIZ string containing the name of the network service being controlled.
- *opcode* is a value indicating the action to perform on the service, defined in SERVICE.H as follows:

Manifest/Bits	Value	Meaning
SERVICE_CTRL_INTERROGATE	0	Interrogate service status.
SERVICE_CTRL_PAUSE	1	Pause service.
SERVICE_CTRL_CONTINUE	2	Continue service.
SERVICE_CTRL_UNINSTALL	3	Stop service.
4-255		Reserved.

- *arg* is a value that indicates the service-specific operation to perform. *arg* values for each service are defined in SERVICE.H. The requester pause and continue commands include the following options:

Manifest	Value	Meaning
SERVICE_CTRL_REDIR_DISK	1	Disk resource.
SERVICE_CTRL_REDIR_PRINT	2	Print resource.
SERVICE_CTRL_REDIR_COMM	4	Serial device.

- *buf* points to the *service_info_2* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_ServiceTableLocked	2180	The service does not respond to control actions.
NERR_ServiceNotInstalled	2184	The service has not been started.
NERR_ServiceCtlTimeout	2186	The service is not responding to the control function.

Manifest	Value	Meaning
NERR_ServiceCtlBusy	2187	The service control is busy.
NERR_ServiceNotCtrl	2189	The service cannot be controlled in its present state.
NERR_ServiceKillProc	2190	The service was ended abnormally.
NERR_ServiceCtlNotValid	2191	The requested pause or stop is not valid for this service.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFlagProcess
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetInfoSeg
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear
- DosSemWait.

Remarks

If the operation requested by the control *opcode* takes a long time to complete, the status and code values that NetServiceControl returns may be intermediate. Thus, for long-running operations, an application should issue successive calls to NetServiceControl to verify that the operation has completed.

NetServiceControl acts only on services that are started. If a service is in the UNINSTALLED, UNINSTALL_PENDING, or INSTALL_PENDING state, NetServiceControl returns the NERR_ServiceCtlNotValid error code. There is one exception to this rule. An application can pass the *opcode* parameter with the value 0 (interrogation) to query the last known state of a stopped service. (If a service has never been started, NetServiceControl returns the NERR_ServiceNotInstalled error code.)

Services can be written to recognize a particular set of opcodes, as appropriate.

Related Information

For information on:

- Listing the services started on a server—See "NetServiceEnum" on page 3-314.
- Updating status and code information for a service—See "NetServiceStatus" on page 3-323.

NetServiceEnum

The NetServiceEnum (DOS) function retrieves information about all network services that are started.

Syntax

```
#include <netcons.h>
#include <service.h>

unsigned far pascal
NetServiceEnum(servername, level, buf, buflen,
               entriesread, totalentries)
char far *            servername;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  entriesread;
unsigned short far *  totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail (0, 1, or 2) requested for the *service_info* data structure.
- *buf* points to the *service_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2101	The device driver is not started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_ServiceTableLocked	2180	The service does not respond to control actions.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

The NetServiceEnum function returns information only on services that are in the start state.

Related Information

For information on updating the status and code information for a network service, see "NetServiceStatus" on page 3-323.

NetServiceGetInfo

The NetServiceGetInfo function retrieves information about a particular network service that is started.

Syntax

```
#include <netcons.h>
#include <service.h>

unsigned far pascal
NetServiceGetInfo (servername, service, level,
                   buf, buflen, totalavail)
const char far *      servername;
const char far *      service;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *service* points to an ASCIIZ string containing the name of the network service for which information is being requested.
- *level* specifies the level of detail (0, 1, or 2) requested for the returned *service_info* data structure.
- *buf* points to the *service_info* data structure.
- *buflen* tells the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2101	The device driver is not started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_ServiceTableLocked	2180	The service does not respond to control actions.
NERR_ServiceNotInstalled	2184	The service has not been started.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

The NetServiceGetInfo function returns the NERR_Success code for services that are not started (SERVICE_UNINSTALLED). If a service is stopped, an application can examine the available data of a server using NetServiceControl.

NetServiceGetInfo function is similar to the NetServiceControl function passed with the INTERROGATE opcode. However, NetServiceGetInfo does not interrogate the service; it only retrieves the status that the service last posted.

Related Information

For information on controlling the operations of a network service, see "NetServiceControl" on page 3-310.

NetServiceInstall

The NetServiceInstall (admin, DOS) function starts a network service.

Syntax

```
#include <netcons.h>
#include <service.h>

unsigned far pascal
NetServiceInstall(servername, service, cmdargs, buf, buflen)
char far *      servername;
char far *      service;
char far *      cmdargs;
char far *      buf;
unsigned short  buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *service* points to an ASCIIZ string containing the name of the network service to start.
- *cmdargs* points to an ASCIIZ string containing the command parameters for service. *cmdargs* can be a NULL pointer or can point to a series of ASCIIZ string parameters ended by a NULL ending character in one of the following forms:

  ```
  param:value\0          >- ASCIIZ parameter
  param\0                     >- ASCIIZ parameter
  param=value\0               >- ASCIIZ parameter
  \0                          >- Null parameter ends list
  ```

  cmdargs parameters are merged with service component parameters from the IBMLAN.INI file and passed to the service program.
- *buf* points to the *service_info_2* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_PATH_NOT_FOUND	3	The path was not found.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_LanIniError	2131	An error occurred when opening or reading the IBMLAN.INI file.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_LineTooLong	2149	A line in the IBMLAN.INI file is too long.
NERR_ServiceTableLocked	2180	The service does not respond to control actions.
NERR_ServiceTableFull	2181	The service table is full.
NERR_ServiceInstalled	2182	The requested service has already been started.
NERR_ServiceEntryLocked	2183	The service does not respond to control actions.
NERR_BadServiceName	2185	The service name is invalid.
NERR_ServiceCtlTimeout	2186	The service is not responding to the control function.

Manifest	Value	Meaning
NERR_ServiceCtlBusy	2187	The service control is busy.
NERR_BadServiceProgName	2188	The IBMLAN.INI file contains an invalid service program name.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocShrSeg-[ERROR_ALREADY_EXISTS]
- DosChgFilePtr
- DosExecPgm
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosOpen[-ERROR_OPEN_FAILED]
- DosRead
- DosSemClear
- DosSemWait-[ERROR_SEM_TIMEOUT]
- DosStartSession.

Remarks

The name of the service is found in the IBMLAN.INI file. The executable file name of the service is matched to a corresponding entry in the SERVICES component of the IBMLAN.INI file. Any relative file path name supplied for service is assumed relative to the OS/2 LAN Requester/Server root directory (\IBMLAN\).

Related Information

For information on:

- Controlling network services—See "NetServiceControl" on page 3-310.
- Listing available servers—See "NetServerEnum2" on page 3-289.

NetServiceStatus

The NetServiceStatus function sets status and code information for a network service.

Syntax

```
#include <netcons.h>
#include <service.h>

unsigned far pascal
NetServiceStatus(buf, buflen)
const char far *  buf;
unsigned short    buflen;
```

where:

- *buf* points to the *service_status* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
NERR_NetNotStarted	2101	The device driver is not started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_ServiceTableLocked	2180	The service does not respond to control actions.
NERR_ServiceNotInstalled	2184	The service has not been started.

Other error return codes may be returned from the following OS/2 functions:

- DosGetInfoSeg
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND].

Remarks

Service applications must call NetServiceStatus to update their status and code tables each time their status changes.

If a non-service application (one not started by a call to NetServiceInstall) calls NetServiceStatus, then NetService returns the NERR_ServiceNotInstalled error code.

Related Information

For information on controlling the operation of a network service, see "NetServiceControl" on page 3-310.

Session Category

NetSessionDel (*admin, server, DOS*)—See "NetSessionDel" on page 3-328.

NetSessionEnum (*partially admin, server, DOS*)—See "NetSessionEnum" on page 3-331.

NetSessionGetInfo (*partially admin, server, DOS*)—See "NetSessionGetInfo" on page 3-334.

The functions in the Sessions category control network sessions established between requesters and servers. They are used with the SHARES.H and NETCONS.H include files.

Description

A session is literally a path between a requester and a server. A requester begins a session with a server the first time it requests to connect to a shared resource on the server. Any further connections from the requester to the other shared resources on the same server do not create another session; multiple connections can be serviced on one session.

To end a session, an application calls the NetSessionDel function. This action deletes all current connections between the requester and the server.

The NetSessionEnum function returns information about all sessions established with a server.

To obtain information about a particular session, an application calls the NetSessionGetInfo function.

Data Structures

The *level* parameter controls the level of information that the NetSessionEnum and NetSessionGetInfo functions return.

Session Information (Level 0)

```
struct session_info_0 {
   char far * sesi0_cname;
};
```

where:

- *sesi0_cname* points to an ASCIIZ string containing the computer name of the requester that established the session.

Session Information (Level 1)

```
struct session_info_1 {
  char far *     sesi1_cname;
  char far *     sesi1_username;
  unsigned short sesi1_num_cons;
  unsigned short sesi1_num_opens;
  unsigned short sesi1_num_users;
  unsigned long  sesi1_sess_time;
  unsigned long  sesi1_idle_time;
  unsigned long  sesi1_user_flags;
};
```

where:

- *sesi1_cname* points to an ASCIIZ string containing the computer name of the requester that established the session.
- *sesi1_username* points to an ASCIIZ string containing the name of the user who established the session.
- *sesi1_num_conns* indicates the number of connections that have been made during the session.
- *sesi1_num_opens* indicates the number of files, devices, and pipes that have been opened during the session.
- *sesi1_num_users* specifies the number of sessions that are established between the server and the requester.
- *sesi1_sess_time* indicates the number of seconds a session has been active.
- *sesi1_idle_time* indicates the number of seconds a session has been idle.
- *sesi1_user_flags* indicates the manner in which the user established the session. The bit mask for *sesi1_user_flags* is defined in SHARES.H as follows:

Manifest	Value	Meaning
SESS_GUEST	1	*sesi1_username* established the session using a GUEST account.
SESS_NOENCRYPTION	2	*sesi1_username* established the session without using password encryption.

Session Information (Level 2)

```
struct session_info_2 {
  char far *     sesi2_cname;
  char far *     sesi2_username;
  unsigned short sesi2_num_conns;
  unsigned short sesi2_num_opens;
  unsigned short sesi2_num_users;
  unsigned long  sesi2_sess_time;
  unsigned long  sesi2_idle_time;
  unsigned long  sesi2_user_flags;
  char far *     sesi2_cltype_name;
};
```

where:

- *sesi2_cname* points to an ASCIIZ string containing the computer name of the requester that established the session.
- *sesi2_username* points to an ASCIIZ string containing the name of the user who established the session.
- *sesi2_num_conns* indicates the number of connections that have been made during the session.
- *sesi2_num_opens* indicates the number of files, devices, and pipes that have been opened during the session.
- *sesi2_num_users* specifies the number of sessions that are established between the server and the requester.
- *sesi2_sess_time* indicates the number of seconds a session has been active.
- *sesi2_idle_time* indicates the number of seconds a session has been idle.
- *sesi2_user_flags* indicates the manner in which the user established the session. The bit mask for *sesi2_user_flags* is defined in SHARES.H as follows:

Manifest	Value	Meaning
SESS_GUEST	1	*sesi2_username* established the session using a GUEST account
SESS_NOENCRYPTION	2	*sesi2_username* established the session without using password encryption.

- *sesi2_cltype_name* specifies the type of client that established the session. The defined types are as follows:

Type	Meaning
Down Level	Old clients; for example, PLCP
DOS LM 1.0	DOS LAN Manager 1.0 clients
DOS LM 2.0	DOS LAN Manager 2.0 clients
OS/2 LS 1.0	OS/2 LAN Server 1.0 clients
OS/2 LS 1.2	OS/2 LAN Server 1.2 clients

Session Information (Level 10)

```
struct session_info_10 {
  char far *     sesi10_cname;
  char far *     sesi10_username;
  unsigned long  sesi10_sess_time;
  unsigned long  sesi10_idle_time;
};
```

where:

- *sesi10_cname* points to an ASCIIZ string containing the computer name of the requester that established the session.
- *sesi10_username* points to an ASCIIZ string containing the name of the user who established the session.

- *sesi10_sess_time* indicates the number of seconds a session has been active.
- *sesi10_idle_time* indicates the number of seconds a session has been idle.

NetSessionDel

The NetSessionDel (admin, server, DOS) function ends a session between a requester and a server.

Syntax

```
#include <netcons.h>
#include <shares.h>

unsigned far pascal
NetSessionDel(servername, workstation, reserved)
char far *  servername;
char far *  workstation;
short       reserved;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *workstation* points to an ASCIIZ string containing the name of the requester that established the session being discontinued.
- *reserved* must be 0.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.

Manifest	Value	Meaning
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_NoSuchServer	2460	The server ID is not valid.
NERR_NoSuchSession	2461	The session ID is not valid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetProcAddr
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosLoadModule
- DosSemClear.

Remarks

All connections established by way of the session are disconnected when the session is deleted, and any files that were opened by way of the session are closed. Data may be lost if any process on the requester is communicating with the server when NetSessionDel is called.

Related Information

For information on retrieving the status of the session of a server, see "NetSessionGetInfo" on page 3-334.

NetSessionEnum

The NetSessionEnum (partially admin, server, DOS) function provides information on all current sessions to a server. Users with less than administrative privileges receive session information at level 0 or level 10.

Syntax

```
#include <netcons.h>
#include <shares.h>

unsigned far pascal
NetSessionEnum(servername, level, buf, buflen,
               entriesread, totalentries)
char far *           servername;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far * entriesread;
unsigned short far * totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (0, 1, 2 or 10) requested for the returned *session_info* data structure.
- *buf* points to the returned data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Related Information

For information on:

- Deleting a session—See "NetSessionDel" on page 3-328.
- Listing all sessions redirected to a resource—See "NetConnectionEnum" on page 3-78.

NetSessionGetInfo

The NetSessionGetInfo (partially admin, server, DOS) function retrieves information about a session established between a particular requester and server. Users with less than administrative privileges receive session information at level 0 or level 10.

Syntax

```
#include <netcons.h>
#include <shares.s>

unsigned far pascal
NetSessionGetInfo(servername, workstation, level,
                  buf, buflen, totalavail)
char far *           servername;
char far *           workstation;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *workstation* points to an ASCIIZ string containing the name of the requester whose session is to be monitored.
- *level* specifies the level of detail (0, 1, 2, or 10) requested for the returned *session_info* data structure.
- *buf* points to the *session_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_ClientNameNotFound	2312	A session does not exist with that computer name.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Related Information

For information on listing all sessions redirected to a resource, see "NetConnectionEnum" on page 3-78.

Share Category

NetShareAdd (*admin, server, DOS*)—See "NetShareAdd" on page 3-340.

NetShareCheck (*server, DOS*)—See "NetShareCheck" on page 3-344.

NetShareDel (*admin, server, DOS*)—See "NetShareDel" on page 3-347.

NetShareEnum (*partially admin, server, DOS*)—See "NetShareEnum" on page 3-350.

NetShareGetInfo (*partially admin, server, DOS*)—See "NetShareGetInfo" on page 3-353.

NetShareSetInfo (*admin, server, DOS*)—See "NetShareSetInfo" on page 3-356.

The functions in the Share category control shared resources. They are used with the SHARES.H, ACCESS.H, and NETCONS.H include files.

Description

Share is the term applied to the local device of a server (such as a disk drive, print device, or named pipe) that other applications on the network can access. A unique *netname* is assigned to each shared resource to enable remote users and applications to refer to the share, rather than the local device name of the share.

The first step for allowing remote users and applications to access a server resource is to share the resource, giving it a netname. This is done with the NetShareAdd function, which adds a share to a server. The function requires information about the resource type.

On a server, NetShareAdd requires only a netname and a local device name to share a resource. A user or application must have an account on the server to access the resource. For information on setting up user accounts in the user accounts subsystem (UAS) database, see "User Category" on page 3-382. For information on assigning permissions, see "Access Permission Category" on page 3-2.

DOS Considerations

Under DOS, the functions can be executed only on a remote server. Attempting to execute the functions on a local requester returns NERR_RemoteOnly.

Data Structures

Share information can be returned by the NetShareEnum and NetShareGetInfo functions at one of three levels of detail specified by the level parameter (values 0, 1, or 2) NetShareAdd requires level 2 of detail. NetShareSetInfo can be called with level 1 or 2. NetShareCheck and NetShareDel do not use or return data structures.

The following data structures are associated with level values 0, 1, and 2.

Share Information (Level 0)

```
struct share_info_0 {
  char shi0_netname[NNLEN+1];
};
```

where:

- *shi0_netname* is an ASCIIZ string containing the netname of a resource.

Share Information (Level 1)

```
struct share_info_1 {
  char           shi1_netname[NNLEN+1];
  char           shi1_pad1;
  unsigned short shi1_type;
  char far *     shi1_remark;
};
```

where:

- *shi1_netname* is an ASCIIZ string containing the netname of a resource.
- *shi1_pad1* WORD-aligns the data structure components.
- *shi1_type* is one of four values indicating the type of share, as defined in SHARES.H:

Manifest	Value	Share Type
STYPE_DISKTREE	0	Disk drive.
STYPE_PRINTQ	1	Spooler queue.
STYPE_DEVICE	2	Serial device.
STYPE_IPC	3	Interprocess communication (IPC).

- *shi1_remark* points to an ASCIIZ string containing an optional comment about the shared resource.

Share Information (Level 2)

```
struct share_info_2 {
  char           shi2_netname[NNLEN+1];
  char           shi2_pad1;
  unsigned short shi2_type;
  char far *     shi2_remark;
  unsigned short shi2_permissions;
  unsigned short shi2_max_uses;
  unsigned short shi2_current_uses;
  char far *     shi2_path;
  char           shi2_passwd[SHPWLEN+1];
  char           shi2_pad2;
};
```

where:

- *shi2_netname* is an ASCIIZ string containing the netname of a resource.
- *shi2_pad1* WORD-aligns the data structure components.
- *shi2_type* is one of four values indicating the type of share, as defined in SHARES.H as follows:

Manifest	Value	Meaning
STYPE_DISKTREE	0	Disk drive.
STYPE_PRINTQ	1	Spooler queue.
STYPE_DEVICE	2	Serial device.
STYPE_IPC	3	Interprocess communication (IPC).

Note: The *shi2_type* value affects the requirements for certain other *share_info_2* components when the NetShareAdd function is called. See "Remarks" in the function reference section for details.

- *shi2_remark* points to an ASCIIZ string containing an optional comment about the shared resource.
- *shi2_permissions* is reserved.
- *shi2_max_uses* gives the maximum number of concurrent connections that the shared resource can accommodate (unlimited if the *shi2_max_uses* value is -1).
- *shi2_current_uses* indicates the number of connections that are currently made to the resource.
- *shi2_path* points to an ASCIIZ string containing the local path name of the shared resource. For disks, *shi2_path* is the path being shared. For spooler queues, *shi2_path* is the name of the spooler queue being shared. For serial device queues, *shi2_path* is a string of one or more communication device names separated by spaces (for example, COM1 COM2 COM6).
- *shi2_passwd* is a reserved field and it must be NULL.
- *shi2_pad2* WORD-aligns the data structure components.

Related Information

For information on:

- User accounts—See "User Category" on page 3-382.
- Access permissions—See "Access Permission Category" on page 3-2.

NetShareAdd

The NetShareAdd (admin, server, DOS) function shares the resource of a server.

Syntax

```
#include <netcons.h>
#include <shares.h>
#include <access.h>

unsigned far pascal
NetShareAdd(servername, level, buf, buflen)
char far *      servername;
short           level;
char far *      buf;
unsigned short  buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (2) provided by the *share_info_2* data structure.
- *buf* points to the *share_info_2* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_FILENAME_EXCED_RANGE	206	The file name is longer than 8 characters or the extension is longer than 3 characters.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_UnknownServer	2103	The server cannot be located.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_UnknownDevDir	2116	The device or directory does not exist.
NERR_RedirectedPath	2117	The operation is invalid on a redirected device.
NERR_DuplicateShare	2118	The name has already been shared.
NERR_NoRoom	2119	The server is currently out of the requested resource.
NERR_TooManyItems	2121	The requested add of item exceeds maximum allowed.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_QNotFound	2150	The printer queue does not exist.

Manifest	Value	Meaning
NERR_DeviceShareConflict	2318	This device cannot be shared as both a spooled and a non-spooled device.
NERR_BadDevString	2340	This list of devices is invalid.
NERR_BadDev	2341	The requested device is invalid.
NERR_InUseBySpooler	2342	This device is already in use by the spooler.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CantType	2357	The type of input cannot be determined.

Other error return codes may be returned from the following OS/2 functions:

- DosDevIOCtl
- DosFsRamSemClear
- DosFsRamSemRequest
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NIOCGETASGLIST)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetProcAddr
- DosGetShrSeg
- DosLoadModule
- DosPrintDestGetInfo[-NERR_SpoolerNotLoaded, NERR_DestNotFound,ERROR_MORE_DATA]
- DosSemClear
- DosSemRequest
- redir.GetNetInitPath
- redir.NIOCGETASGLIST1_1.

Remarks

Depending on the type of share specified by the value in the *shi2_type* component of the *share_info_2* data structure, other components in the data structure must be specified as follows:

Manifest	Value	Component Requirements
STYPE_DISKTREE	0	*shi2_path* must specify a file system path name.
STYPE_PRINTQ	1	*shi2_netname* must specify the name of an existing spooler queue. It is recommended that the netname and the path be the same; otherwise, the result is unpredictable.
STYPE_DEVICE	2	*shi2_path* must be passed as a NULL pointer or point to a list of print destinations, separated by spaces. The list must be the same as those specified for the spooler queue.
STYPE_IPC	3	*shi2_netname* must specify a shared interprocess communication resource, and *shi2_path* must point to a NULL string.

NetShareAdd ignores the value specified in the *shi2_current_uses* component of *share_info_2*.

Related Information

For information on:

- Removing a list of shareable resources—See "NetShareDel" on page 3-347.
- Listing the shareable resources of a server—See "NetShareEnum" on page 3-350.

NetShareCheck

The NetShareCheck (server, DOS) function queries as to whether a server is sharing a device.

Syntax

```
#include <netcons.h>
#include <shares.h>
#include <access.h>

unsigned far pascal
NetShareCheck(servername, devname, type)
char far *           servername;
char far *           devname;
unsigned short far *  type;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *devname* points to an ASCIIZ string containing the name of the device to be checked.
- *type* points to an unsigned short integer indicating the type of shared device, as defined in SHARES.H as follows:

Manifest	Value	Meaning
STYPE_DISKTREE	0	Disk drive.
STYPE_PRINTQ	1	Spooler queue.
STYPE_DEVICE	2	Serial device.
STYPE_IPC	3	Interprocess communication device.

The returned *type* value is valid only if NetShareCheck is successful.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.

Manifest	Value	Meaning
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NetNameNotFound	2310	This shared resource does not exist.
NERR_DeviceNotShared	2311	This device is not shared.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)

- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

The NetShareCheck function returns successfully if a specified device is in the routing list of a spooler queue or serial device queue.

Related Information

For information on:

- Reconfiguring the shareable resource of a server—See "NetShareSetInfo" on page 3-356.
- Retrieving the status of a shared resource—See "NetShareGetInfo" on page 3-353.

NetShareDel

The NetShareDel (admin, server, DOS) function deletes a net name from the list of shared resources of a server.

Syntax

```
#include <netcons.h>
#include <shares.h>
#include <access.h>

unsigned far pascal
NetShareDel(servername, netname, reserved)
char far *      servername;
char far *      netname;
unsigned short  reserved;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *netname* points to an ASCIIZ string specifying the netname to be deleted.
- *reserved* is 0.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.

Manifest	Value	Meaning
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NetNameNotFound	2310	This shared resource does not exist.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- AltSrvShareDel
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetProcAddr
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosLoadModule
- DosSemClear.

Remarks

When the NetShareDel function deletes a netname, all connections to the shared resource are disconnected.

Related Information

For information on:

- Adding a share on a server—See "NetShareAdd" on page 3-340.
- Listing all connections to a shared resource—See "NetConnectionEnum" on page 3-78.
- Listing the shareable resources of a server—See "NetShareEnum" on page 3-350.

NetShareEnum

The NetShareEnum (partially admin, server, DOS) function retrieves share information about each shared resource on a server.

Syntax

```
#include <netcons.h>
#include <shares.h>
#include <access.h>

unsigned far pascal
NetShareEnum(servername, level, buf, buflen,
             entriesread, totalentries)
char far *            servername;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  entriesread;
unsigned short far *  totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (0, 1, or 2) returned in the *share_info* data structure.
- *buf* points to the *share_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

If the *level* parameter of NetShareEnum is passed with the value 2, the calling application must have administrative privileges on the server.

Related Information

For information on retrieving information about a particular shared resource, see "NetShareGetInfo" on page 3-353.

NetShareGetInfo

The NetShareGetInfo (partially admin, server, DOS) retrieves information about a particular shared resource on a server.

Syntax

```
#include <netcons.h>
#include <shares.h>
#include <access.h>

unsigned far pascal
NetShareGetInfo(servername, netname, level, buf,
                buflen, totalavail)
char far *            servername;
char far *            netname;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *netname* points to an ASCIIZ string containing the name of the share of interest.
- *level* specifies the level of detail (0, 1, or 2) returned in the *share_info* data structure.
- *buf* points to the *share_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NetNameNotFound	2310	This shared resource does not exist.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)

- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

If the *level* parameter of NetShareGetInfo is passed with the value 2, the calling application must have administrative privileges on the server.

Related Information

For information on reconfiguring a shareable server resource, see "NetShareSetInfo" on page 3-356.

NetShareSetInfo

The NetShareSetInfo (admin, server, DOS) function sets the parameters of a shared resource.

Syntax

```
#include <netcons.h>
#include <shares.h>
#include <access.h>

unsigned far pascal
NetShareSetInfo(servername, netname, level, buf, buflen, parmnum)
char far *      servername;
char far *      netname;
short           level;
char far *      buf;
unsigned short  buflen;
unsigned short  parmnum;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *netname* points to an ASCIIZ string containing the netname of the resource to be set.
- *level* specifies the level of detail (1 or 2) provided by the *share_info* data structure.
- *buf* points to the data structure if *parmnum* is zero. Otherwise, *buf* points to the specific data component that will be changed.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* specifies whether a specific component of the *share_info* data structure is being set, or the entire data structure. If *parmnum* is 0, then *buf* must contain a complete *share_info* data structure. Otherwise, *parmnum* must pass the ordinal position value for one of the following *share_info_2* data structure components, as defined in SHARES.H as follows:

Manifest	Value	Component
SHI_REMARK_PARMNUM	4	*shi1_remark* or *shi2_remark*
SHI_MAX_USES_PARMNUM	6	*shi2_max_uses*

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.

Manifest	Value	Meaning
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_NAME	123	There is an incorrect character or incorrectly formed file system name.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_NoRoom	2119	The server is currently out of the requested resource.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.

Manifest	Value	Meaning
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_NameNotFound	2273	The message alias cannot be found on the local area network.
NERR_NetNameNotFound	2310	This shared resource does not exist.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Related Information

For information on retrieving the status of a shareable server resource, see "NetShareGetInfo" on page 3-353.

Spooler Category

For information on the following functions, see the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference.*

SplQmAbort

SplQmClose

SplQmEndDoc

SplQmOpen

SplQmStartDoc

SplQmWrite.

The functions in the Spooler category provide the application access to spooler queue manager operations. These functions are provided by the base operating system and supported by the LAN Server across the network.

Description

For detailed descriptions of the above functions, refer to the *IBM Operating System/2 Technical Reference Version 1.2 Programming Reference.*

Statistics Category

NetStatisticsGet2 (*admin, DOS*)—See "NetStatisticsGet2" on page 3-365.

The functions in the Statistics category retrieve and clear the operating statistics for requesters and servers. They are used with the NETSTATS.H and NETCONS.H include files.

Description

The OS/2 LAN Requester/Server accumulates a set of operating statistics for requesters and servers from the time that a *requester* or *server* service is started. The NetStatisticsClear function can be called to clear the statistics. To retrieve the statistics, call the NetStatisticsGet2 function.

Data Structures

NetStatisticsGet2 function returns the statistics in the following data structures:

Data Structure of Requester Statistics

```
struct stat_workstation_0 {
  unsigned long stw0_start;
  unsigned long stw0_numNCB_r;
  unsigned long stw0_numNCB_s;
  unsigned long stw0_numNCB_a;
  unsigned long stw0_fiNCB_r;
  unsigned long stw0_fiNCB_s;
  unsigned long stw0_fiNCB_a;
  unsigned long stw0_fcNCB_r;
  unsigned long stw0_fcNCB_s;
  unsigned long stw0_fcNCB_a;
  unsigned long stw0_sesstart;
  unsigned long stw0_sessfailcon;
  unsigned long stw0_sessbroke;
  unsigned long stw0_uses;
  unsigned long stw0_usefail;
  unsigned long stw0_autorec;
  unsigned long stw0_bytessent_r_high;
  unsigned long stw0_bytessent_r_low;
  unsigned long stw0_bytesrcvd_r_high;
  unsigned long stw0_bytesrcvd_r_low;
  unsigned long stw0_bytessent_s_high;
  unsigned long stw0_bytessent_s_low;
  unsigned long stw0_bytesrcvd_s_high;
  unsigned long stw0_bytesrcvd_s_low;
  unsigned long stw0_bytessent_a_high;
  unsigned long stw0_bytessent_a_low;
  unsigned long stw0_bytesrcvd_a_high;
  unsigned long stw0_bytesrcvd_a_low;
  unsigned long stw0_reqbufneed;
  unsigned long stw0_bigbufneed;
};
```

where:

- *start* is the time that statistics collection started. This field indicates the date or time that the statistics were last cleared; that is, it indicates the time period over which the returned statistics were collected.

- *numNCB* fields indicate the number of network control blocks (NCBs) issued from each source and include failed NCBs. To get the total successful NCBs issued it is necessary to subtract the numbers of failed NCBs. These numbers are held as follows in the *fiNCB* and *fcNCB* fields:
  - *numNCB_r* is the number of NCBs issued (redirector)
  - *numNCB_s* is the number of NCBs issued (server)
  - *numNCB_a* is the number of NCBs issued (application).
- *fiNCB* fields indicate the number of NCBs that failed at the time they were issued, for whatever reason. These NCBs are still included in the "total issued" count, as follows:
  - *fiNCB_r* is the number of NCBs that failed issue (redirector)
  - *fiNCB_s* is the number of NCBs that failed issue (server)
  - *fiNCB_a* is the number of NCBs that failed issue (application).
- *fcNCB* fields indicate the number of NCBs that failed after issue, at or before completion. These NCBs are still included in the "total issued" count, as follows:
  - *fcNCB_r* is the number of NCBs that failed completion (redirector)
  - *fcNCB_s* is the number of NCBs that failed completion (server)
  - *fcNCB_a* is the number of NCBs that failed completion (application).
- *sessstart* is the number of requester sessions started.
- *sessfailcon* is the number of requester session failures to connect, except those that failed due to "name not found."
- *sessbroke* is the number of failures of requester sessions, after the session was established.
- *uses* is the number of requester uses.
- *usefail* is the number of requester use failures. This is a count of failed tree-connects, when a server is found but the resources were not found.
- *autorec* is the number of requester auto-connects.
- The following 12 fields form six *quad-WORDs*, which contain very large counters. The *high DWORD* of each is the value divided by 2^32, while the *low DWORD* is the value modulo 2^32. A *quad-WORD* is a data area with the size twice as large as a double word.

  These fields count total bytes in all NCBs sent and received for all three categories. Server information is included to provide an accurate total.

  Note: For all the NCB-related and bytes-count counters:

  The suffix *_r* indicates redirector. These NCBs are issued by the redirector as part of the normal process of maintaining remote network connections.

  Those with the suffix *_s* are server-related, sent by the redirector on behalf of the server to maintain shared resource connections.

Those with the suffix *_a* are application-generated NCBs, which may be caused by applications calling NetBiosSubmit, use of second-class mailslots, server announcements (sending and receiving), and so on.

- *bytessent_r_hi* is the number of requester bytes sent to the network (high DWORD).
- *bytessent_r_lo* is the number of requester bytes sent to the network (low DWORD).
- *bytesrcvd_r_hi* is the number of requester bytes received from the network (high DWORD).
- *bytesrcvd_r_lo* is the number of requester bytes received from the network (low DWORD).
- *bytessent_shi* is the number of server bytes sent to the network (high DWORD).
- *bytessent_s_lo* is the number of server bytes sent to the network (low DWORD).
- *bytesrcvd_s_hi* is the number of requester bytes received from the network (high DWORD).
- *bytesrcvd_s_lo* is the number of requester bytes received from the network (low DWORD).
- *bytessent_a_hi* is the number of application bytes sent to the network (high DWORD).
- *bytessent_a_lo* is the number of application bytes sent to the network (low DWORD).
- *bytesrcvd_a_hi* is the number of application bytes received from the network (high DWORD).
- *bytesrcvd_a_lo* is the number of application bytes received from the network (low DWORD).

- *reqbufneed* is the number of times that the requester required a request buffer but failed to allocate one. This indicates that the parameters of the requester may need adjustment.
- *bigbufneed* is the number of times the requester required a big buffer but failed to allocate one. This indicates that the parameters of the requester may need adjustment.

Note: A value of -1 or 0xFFFFFFFF for any field means that information is not available. A value of -2 or 0xFFFFFFFE means that the field has overflowed.

Data Structure of Server Statistics

```
struct stat_server_0 {
  unsigned long sts0_start;
  unsigned long sts0_fopens;
  unsigned long sts0_devopens;
  unsigned long sts0_jobsqueued;
  unsigned long sts0_sopens;
  unsigned long sts0_stimedout;
  unsigned long sts0_serrorout;
  unsigned long sts0_pwerrors;
  unsigned long sts0_permerrors;
  unsigned long sts0_syserrors;
  unsigned long sts0_bytessent_high;
  unsigned long sts0_bytessent_low;
  unsigned long sts0_bytesrcvd_high;
  unsigned long sts0_bytesrcvd_low;
  unsigned long stw0_avresponse;
  unsigned long stw0_reqbufneed;
  unsigned long stw0_bigbufneed;
};
```

where:

- *start* is the time statistics collection started. This field indicates the date and time that the statistics were last cleared; that is, it indicates the time period over which the returned statistics were collected.
- *fopens* is the number of server file opens. This includes opens of named pipes.
- *devopens* is the number of server device opens.
- *jobsqueued* is the number of server print jobs spooled.
- *sopens* is the number of server session starts.
- *stimedout* is the number of server session auto-disconnects.
- *serrorout* is the number of server sessions errored out.
- *pwerrors* is the number of server password violations.
- *permerrors* is the number of server access permission errors.
- *syserrors* is the number of server system errors.
- The following 4 fields form two *quad-WORDs* that contain very large counters. The *high DWORD* of each is the value divided by 2^32, while the *low DWORD* is the value modulo 2^32.
  - *bytessent_high* is the number of server bytes sent to the network (high DWORD)
  - *bytessent_low* is the number of server bytes sent to the network (low DWORD)
  - *bytesrcvd_high* is the number of server bytes received from the network (high DWORD)
  - *bytesrcvd_low* is the number of server bytes received from the network (low DWORD).
- *avresponse* is the average server response time in milliseconds.

- *reqbufneed* is the number of times the server required a request buffer but failed to allocate one. This indicates that the parameters of the server may need adjustment.
- *bigbufneed* is the number of times the server required a big buffer but failed to allocate one. This indicates that the parameters of the server may need adjustment.

Note: A value of -1 or 0xFFFFFFFF for any field means that information is not available. A value of -2 or 0xFFFFFFFE means that the field has overflowed.

NetStatisticsGet2

The NetStatisticsGet2 (admin, DOS) function retrieves and optionally clears operating statistics for a service.

Syntax

```
#include <netcons.h>
#include <netstats.h>

unsigned far pascal
NetStatisticsGet2(servername, servicename, reserved, level, options,
                  buf, buflen, totalavail)
char far *            servername;
char far *            servicename;
unsigned long         reserved;
short                 level;
unsigned long         options
char far *            buf;
unsigned short        buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *servicename* is the ASCIIZ service name for which to get the statistics.

  For the OS/2 LAN Requester/Server version 1.2 program, only SERVER and REQUESTER are allowed for the *servicename*. Other names will produce the ERROR_NOT_SUPPORTED error code. If the server statistics are requested and the server is not running, the error code returned is NERR_ServiceNotInstalled.
- *reserved* must be zero.
- *level* specifies the level of detail (0) returned by the *stat_workstation_0* or the *stat_server_0* data structure.

  The valid values for *level* depend on the value of *servicename*. The current valid value for REQUESTER or SERVER is 0.
- *options* are the options flags.

  The *options* parameter is bitmapped as follows:

Bit	Mask	Symbol	Meaning
0	0x1	STATSOPT_CLR	Clear statistics after retrieval
1-31			Must be zero

  The option to clear the statistics allows automatic Get and Clear operations, which allows an atomic get and clear operation, which allows an application that is compiling cumulative numbers to make sure that no data slips by in the time between the Get operation and the Clear operation.
- *buf* points to the returned *stat_workstation_0* or the *stat_server_0* data structure.

- *buflen* specifies the size (in bytes) of *buf*.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NETWORK_ACCESS_DENIED	65	Network access is denied.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.

Manifest	Value	Meaning
NERR_ServiceNotInstalled	2184	The service has not been started.
NERR_BadServiceName	2185	The service name is invalid.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosSemClear.

Remarks

This API returns a full data structure. As with other GetInfo calls, the error NERR_BuffTooSmall is returned if the supplied buffer is too small for the required data.

If the service name specified is REQUESTER, then *stat_workstation_0* is returned to *buf*. If the service name specified is SERVER, then *stat_server* is returned to *buf*.

Use Category

NetUseAdd (*admin, DOS*)—See "NetUseAdd" on page 3-372.

NetUseDel (*admin, DOS*)—See "NetUseDel" on page 3-375.

NetUseEnum (*admin, DOS*)—See "NetUseEnum" on page 3-378.

NetUseGetInfo (*admin, DOS*)—See "NetUseGetInfo" on page 3-380.

The functions in the Use category examine or control connections (uses) between requesters and servers. Administrative privilege is required to call them remotely. They are used with the USE.H and NETCONS.H include files.

Description

The NetUseAdd function establishes a connection between a local computer and a resource shared on a server by redirecting a NULL or local device name to a shared resource on that remote server. The following types of connections can be made:

- Device name connections, which can only be *explicit*
- Universal naming convention (UNC) connections, which can be *explicit* or *implicit*.

To establish an explicit device name connection, NetUseAdd redirects a local device name to the netname of a remote server resource (*\\servername\netname*). Once a device name connection is made, users or applications can use the remote resource by specifying the local device name.

UNC connections can be *explicit*, created by the NetUseAdd function, or *implicit*, made by way of an OS/2 function (responsible for the connection). Once a UNC connection is established, users or programs can access the remote resource by specifying just the netname of the resource.

To establish an explicit UNC connection, NetUseAdd redirects a NULL device name to the netname of a remote server resource.

To establish an implicit UNC connection, an application passes the netname of the resource to any one of the OS/2 functions that accept netnames (such as the DosOpen function). The UNC name will be understood by the OS/2 function and a connection will be made to the specified netname. All further requests on this connection require the full netname.

Note: Connections are to be distinguished from sessions. A session is a path between a requester and a server, established the first time a requester makes a connection with one of the shared resources of the server. All further connections between the requester and the server are part of this same session until ended by calling the NetSessionDel function.

As an example of how the three types of connections work, consider the following examples:

- Explicit device name connection. Assume an application redirected the local device name d: to the netname *develop\srcdrv* by calling the NetUseAdd function as shown:

  ```
  strncpy (buf.ui1_local, "d:", 3);
  retcode = NetUseAdd ( NULL, 1, buf, BUFLEN );
  ```

  To access files on this resource, an application need only specify the redirected device name and the name of the file, as shown:

  ```
  retcode=system("type d:\\read.me");
  ```

- Explicit UNC connection. Assume an application redirected a NULL device name to the remote resource *develop\srcdrv* by calling the NetUseAdd function, as shown:

  ```
  strncpy (buf.ui1_local, "", 3);
  retcode = NetUseAdd ( NULL, 1, buf, BUFLEN );
  ```

  To display the contents of the READ.ME file on the resource *develop\srcdrv*, an application can also just specify the name of the resource with the following command:

  ```
  retcode=system("type \\\\develop\\srcdrv\\read.me");
  ```

  Note that this does not create a new connection to the resource, as an implicit UNC connection would if no NetUseAdd function was called.

- Implicit UNC connection. An implicit connection is made by a call to an OS/2 function and by passing the remote netname as part of a parameter. For example, the following call to DosOpen establishes an implicit UNC connection to the remote resource *develop\srcdrv* and *opens file.1*:

  ```
  retcode = DosOpen ("\\\\develop
  \\srcdrv\\file.1, ...");
  ```

  A password must be supplied the first time a universal naming convention (UNC) connection is made between a local or NULL device name and a remote resource if the resource is on a server running in user-level security. Further connections to the same server do not require a password because the password is associated with the same session.

DOS Considerations

Under DOS, the functions can be executed only on a remote server. Attempting to execute the functions on a local requester returns NERR_RemoteOnly.

Data Structures

The data structures used by the NetUseAdd, NetUseEnum, and NetUseGet functions are described immediately following the syntax descriptions for each function.

Use Information (Level 0)

```
struct use_info_0 {
char       ui0_local[DEVLEN+1];
char       ui0_pad_1;
char far * ui0_remote;
};
```

where:

- *ui0_local* is an ASCIIZ string specifying the local device name (such as E:, LPT1, or COM1) being redirected to the shared resource.
- *ui0_pad_1* WORD-aligns the data structure components.
- *ui0_remote* points to an ASCIIZ string containing the netname of the remote resource being accessed. The string must be in the form *\\servername\netname*.

Use Information (Level 1)

```
sruct use_info_1 {
char          ui1_local[DEVLEN+1];
char          ui1_pad_1;
char far *    ui1_remote;
char far *    ui1_password;
unsigned char ui1_status;
unsigned char ui1_asg_type;
unsigned char ui1_refcount;
unsigned char ui1_usecount;
};
```

where:

- *ui1_local* is an ASCIIZ string specifying the local device name being redirected to the shared resource.
- *ui1_pad_1* WORD-aligns the data structure components.
- *ui1_remote* points to an ASCIIZ string specifying the UNC name of the remote resource being accessed. The string must be in the form *\\servername\netname*.
- *ui1_password* is a reserved field; it must be NULL.
- *ui1_status* specifies the status of the connection. Seven possible values for *ui1_status* are defined in USE.H, as follows:

Manifest	Value	Meaning
USE_OK	0	Connection valid.
USE_PAUSED	1	Paused by local requester.
USE_SESSLOST	2	Session removed.
USE_DISCONN	2	Connection disconnected
USE_NETERR	3	Network error.
USE_CONN	4	Connection being made.
USE_RECONN	5	Reconnecting.

- *ui1_asg_type* specifies the type of remote resource being accessed. Five types of resources are defined in USE.H, as follows:

Manifest	Value	Meaning
USE_WILDCARD	-1	Matches the type of the share of the server (Wildcards are only used when *uil_local* is a NULL string).
USE_DISKDEV	0	Disk device.
USE_SPOOLDEV	1	Spooled printer.
USE_CHARDEV	2	Serial device.
USE_IPC	3	Interprocess communication (IPC).

The *asg_type* field indicates the sort of device to use the resource "as." It can be any of the previously listed values under *asg_type*, or it can be set to USE_WILDCARD (-1), only for a connection with a NULL local device. A connection that maps a local device to the resource must use one of the other four values defined in *asg_type*.

- *uil_refcount* indicates the number of files, directories, and other processes that are open on the remote resource.
- *uil_usecount* indicates the number of explicit connections (redirection of a local device name) or implicit UNC connections (redirection of a NULL local device name) that are established with the resource.

  If both an explicit and an implicit connection exists between a requester and a resource, the *usecount* of the server is 1 and the *usecount* of the requester is 2.

NetUseAdd

The NetUseAdd (admin, DOS) function establishes a connection between a local or NULL device name and shared resource by redirecting the local or NULL (UNC) device name to the shared resource.

Syntax

```
#include <netcons.h>
#include <use.h>

unsigned far pascal
NetUseAdd(servername, level, buf, buflen)
char far *      servername;
short           level;
char far *      buf;
unsigned short  buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail (1) for the *use_info_1 data* structure.
- *buf* points to the *use_info_1* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_BAD_DEV_TYPE	66	This network device type is incorrect.
ERROR_BAD_NET_NAME	67	This network name cannot be found.
ERROR_ALREADY_ASSIGNED	85	Duplicate redirection.
ERROR_INVALID_PASSWORD	86	The specified password is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.

Manifest	Value	Meaning
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_UseNotFound	2250	The connection cannot be found.
NERR_BadAsgType	2251	This asg_type is invalid.
NERR_DeviceIsShared	2252	This device is already being shared.
NERR_DeviceNotShared	2311	This device is not shared.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_LocalDrive	2415	Drive in local use.

Other error return codes may be returned from the following OS/2 functions:

- DosFsRamSemClear
- DosFsRamSemRequest
- DosFreeSeg
- DosFsAttach
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NIOCSETASGLIST)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg
- DosSemClear
- DosSemRequest
- redir.(NIOCSETASGLIST1_1).

Remarks

A local device name can be redirected to only one shared resource at a time. To establish a new direction for a redirected device, an application must first delete the existing connection by calling the NetUseDel function.

A COM or LPT device can be redirected even if it is already opened. Redirecting the device has no effect on the handle of the device if it is already open. However, subsequent openings will return a handle to the remote device to which the local device was redirected.

When a server is running user-level or share-level security, The *uil_password* needed to access a shared resource can be provided in the following ways:

- A non-NULL string specifies the password
- A NULL pointer forces the OS/2 LAN Requester/Server software to use the same password given to the logon function
- A NULL string indicates that no password is provided.

NetUseAdd ignores the *ui1_status, ui1_refcount,* and *ui1_usecount* components in the *use_info_1* data structure.

If a call to NetUseAdd duplicates an existing connection, the *usecount* component of the *use_info_1* data structure is incremented and the function succeeds, returning NERR_Success.

Related Information

For information on:

- Disconnecting a device from a shared resource—See "NetUseDel" on page 3-375.
- Listing all devices redirected to a shared resource—See "NetUseEnum" on page 3-378.

NetUseDel

The NetUseDel (admin, DOS) function ends a connection between a local or UNC device name and a shared resource.

Syntax

```
#include <netcons.h>
#include <use.h>

unsigned far pascal
NetUseDel(servername, devicename, forceflag)
char far *  servername;
char far *  devicename;
short       forceflag;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *devicename* points to an ASCIIZ string indicating the local device name that was redirected to the shared resource.
- *forceflag* is one of three values specifying three types of disconnection. As defined in USE.H, the following options are available:

Manifest	Value	Meaning
USE_NOFORCE	0	Maintains the connection in a dormant state, decrementing *usecount*. A dormant session can quickly be activated as soon as reconnection is needed, improving system performance.
USE_FORCE	1	Connection is removed only if no file, directory, or drive is opened. *usecount* is decremented (for a local device name connection) and forced to 0 (for a UNC connection).
USE_LOTS_OF_FORCE	2	All files, directories, and drives open on the connection are forced closed.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.

Manifest	Value	Meaning
ERROR_INVALID_DRIVE	15	The specified drive is not valid.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_BAD_DEV_TYPE	66	This network device type is incorrect.
ERROR_BAD_NET_NAME	67	This network name cannot be found.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_PROTECTION_VIOLATION	115	Incorrect user virtual address.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_OpenFiles	2401	There are open files on the connection.

Other error return codes may be returned from the following OS/2 functions:

- DosFsAttach
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NIOCSETASGLIST)
- DosFsCtl(NULLTRANSACT)
- redir.(NIOCSETASGLIST1_1).

Related Information

For information on:

- Listing all local device names redirected to a shared resource—See "NetUseEnum" on page 3-378.
- Redirecting a local device name to a shared resource—See "NetUseAdd" on page 3-372.

NetUseEnum

The NetUseEnum (admin, DOS) function lists all current connections between the local computer and resources on a remote server.

Syntax

```
#include <netcons.h>
#include <use.h>

unsigned far pascal
NetUseEnum(servername, level, buf, buflen,
           entriesread, totalentries)
char far *            servername;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  entriesread;
unsigned short far *  totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *level* specifies the level of detail (0 or 1) returned by the *use_info* data structure.
- *buf* points to the *use_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_DRIVE	15	The specified drive is not valid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NIOCSETASGLIST)
- DosDevIOCtl(NIOCSETASGLIST1_1)
- DosFsCtl(NULLTRANSACT)
- redir.(NIOCSETASGLIST1_1).

Related Information

For information on:

- Listing the shared resources of a server—See "NetShareEnum" on page 3-350.
- Retrieving the status of a local device name—See "NetUseGetInfo" on page 3-380.

NetUseGetInfo

The NetUseGetInfo (admin, DOS) function retrieves information about a connection between a local device name and a shared resource.

Syntax

```
#include <netcons.h>
#include <use.h>

unsigned far pascal
NetUseGetInfo(servername, netname, level,
         buf, buflen, totalavail)
char far *           servername;
char far *           netname;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local computer.
- *netname* points to an ASCIIZ string containing the name of the redirected device name or UNC name for the resource.
- *level* specifies the level of detail (0 or 1) returned in the *use_info* data structure.
- *buf* points to the *use_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_RemoteErr	2127	A remote API error has occurred.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_UseNotFound	2250	The connection cannot be found.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosDevIOCtl(NIOCGETASGLIAT1_1)
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NIOCSETASGLIST)
- DosFsCtl(NULLTRANSACT)
- redir.NIOCSETASGLIST1_1 [ERROR_NO_MORE_FILES].

Remarks

The NetUseGetInfo function returns the *ui1_password* component as a NULL pointer, so users or applications cannot determine the password of another user or application.

Related Information

For information on:

- Listing all connections to a shared resource—See "NetUseEnum" on page 3-378.
- Listing the shared resources of a server—See "NetShareEnum" on page 3-350.

User Category

NetUserAdd (*admin, DOS*)—See "NetUserAdd" on page 3-397.

NetUserDel (*admin, DOS*)—See "NetUserDel" on page 3-401.

NetUserEnum (*partially admin, DOS*)—See "NetUserEnum" on page 3-404.

NetUserGetGroups (*partially admin, DOS*)—See "NetUserGetGroups" on page 3-407.

NetUserGetInfo (*partially admin, DOS*)—See "NetUserGetInfo" on page 3-410.

NetUserModalsGet (*partially admin, DOS*)—See "NetUserModalsGet" on page 3-413.

NetUserModalsSet (*admin, DOS*)—See "NetUserModalsSet" on page 3-416.

NetUserPasswordSet (*DOS*)—See "NetUserPasswordSet" on page 3-419.

NetUserSetGroups (*admin, DOS*)—See "NetUserSetGroups" on page 3-423.

NetUserSetInfo (*partially admin, DOS*)—See "NetUserSetInfo" on page 3-426.

NetUserValidate2 (*local*)—See "NetUserValidate2" on page 3-431.

The functions in the User category control a user's account in the user accounts subsystem (UAS) database. They are used with the ACCESS.H and NETCONS.H include files.

Description

In the UAS, each user or application that accesses the resources must have a user account in the system. The system uses this account to verify that the user or application has permission to connect to any shared resource. A user's account is set up by calling the NetUserAdd function. There are three types of information in a user's account:

- One type can be set only by the administrator
- Another can be set by the owner of the account
- The third can be set only by the system.

The following user account information can be set by the administrator only:

- Account Name—The name used to log on (for example, JSMITH).
- Full Name Field—The full name of the user (for example, John Smith, Jr.). This field can be up to MAXCOMMENTSZ + 1 bytes long. A NULL pointer is treated as a null string.
- Comment—The comment associated with an account (for example, Vice President). It can contain any text and can be up to MAXCOMMENTSZ + 1 bytes.
- Parms—The drive of the user's home directory.
- Home Directory—The network directory (absolute, UNC, or local path) for the user's files. The field length can be up to PATHLEN bytes.
- Account Disabled—When set, no one can log on using this account.

- Privilege Level—An account can have one of three levels of privilege: GUEST, USER, or ADMIN. Privilege level affects the default rights of an account to access resources and to call APIs remotely.
- Logon Script Flag—Must be TRUE for this release.
- Logon Script Path—The path name of the user's logon script. It must be a relative path, and the path is relative to the NETLOGON service SCRIPTS path. This field can be up to PATHLEN bytes long, and a NULL pointer is treated as a null string. the script can be a *.pro*, a *.cmd* (or *.bat*), an *.exe* or a *.com*, or it can have no extension which is defaulted to *.cmd* (or *.bat*) file.
- Account Expiration Date—Logon is disabled after specified date.
- List of Authorized Requesters—Account can only log on from listed PCs; the list may include zero to eight stations. Zero stations means ALL.
- Logon Hours—1 hour granularity, hours specified for each day of the week. A NULL pointer means ALL for NetUserAdd calls but means DONT_CHANGE for NetUserSetInfo calls.
- Home Directory Required—When this bit is set, UAS requires that the Home Directory field must not be NULL.
- Password Not Required—When this bit is set, the account can have a NULL password even if the Minumum Password Length for the system is greater than 0.
- Maximum Storage Allotment—Specifies the limit in KB on the home directory.
- User Can Not Change Password—This flag restricts the user from changing the password on his account.
- Logon server—The name of the server which tracks logon and logoff for the account. It should be NULL for this release. NULL means that the domain controller validates the logon requests.

The following fields can be set by a user (or by an administrator) for the user's own account only:

- Encrypted Password.
- User Comment—A user-settable comment. This field can be up to MAXCOMMENTSZ + 1 bytes long. A NULL pointer is treated as a null string.
- Parms—An ASCIIZ string for application use.
- Country Code—The OS/2 country code for the language choice of the user. This is used by the OS/2 LAN Requester/Server software to generate messages in the appropriate language whenever possible.
- Code Page—The code page of the OS/2 program for national language support.

The following fields are automatically set by the Accounts System and are not directly set by an administrator or user:

- Date Password Last Changed.—This field is ignored by NetUserAdd and NetUserSetInfo calls.
- Recent Passwords—The eight most recent encrypted passwords for this account.

When a user or application requests access to any shared resource, the OS/2 LAN Requester/Server software checks to see whether there is an appropriate account. An application can do this by calling the NetUserValidate2 function to search for a valid account with a particular user name and password combination.

If the account is found, the application then can check to see whether it is currently enabled by calling the NetUserGetInfo function and examining the flags that are set.

The OS/2 LAN Requester/Server software then checks the user's privilege level. Depending on the parameter, the request to access a resource is immediately accepted, if the user has been given administrative privileges, or processing continues.

Administrative privileges grant the broadest access to the domain, giving permission to execute all administrative functions, and complete access to the shared and non-shared resources of a server.

If the user does not have administrative privileges, the OS/2 LAN Requester/Server software checks the resource's access permission record to see whether the user has the proper permissions to use that particular resource (see "Access Permission Category" on page 3-2).

Each time an account that has either administrative or user privileges is established by calling the NetUserAdd function, the OS/2 LAN Requester/Server software automatically adds a new account to the special group. At this time, the user inherits all permissions assigned to that special group.

An application can change a user's current privileges by calling the NetUserSetInfo function. Or it can change the user's permissions by modifying that user's group accounts (see "Group Category" on page 3-110). Individually assigned user permissions take precedence over group permissions assignments. An application can verify which groups a user belongs to by calling the NetUserGetGroups function. This function returns a list of group names.

To find out how to change individual permission settings for members of one of the the special group, see "Access Permission Category" on page 3-2.

When a user account is no longer needed, the NetUserDel function can be called to eliminate the account from the system. Once the account is removed, the user can no longer access the system.

If an account does not have a password, any password (or none) is treated as a match for account validation.

UAS modals can be used for global settings of password policy within the account system database. None of the system-wide password characteristics are enforced for accounts that do not require passwords.

It is recommended that passwords be entered in uppercase characters. The password of a user account is case-sensitive at the API level, and lowercase characters in the password makes the user ID unavailable at the command-line interface and through the user profile management (UPM) full-screen interface.

The user's password is confidential and is not returned when the NetUserEnum or NetUserGetInfo function is called; a string of spaces is substituted for any password that is requested. The password is initially assigned when NetUserAdd is called. NetUserPasswordSet can be used by any user or application to change their own password, if the current password can be supplied. To verify an existing user account with a specified password, call NetUserValidate2. The NetUserSetInfo function can set the password and other components of a user account.

DOS Considerations

The NetUserPasswordSet function controls a user's password account in a domain. NetUserPasswordSet can be executed only on a remote server. Attempting to execute it on a local requester returns NERR_RemoteOnly.

Data Structures

There are three types of data structures in the User category: user account data structures, user modal data structures, and user validation data structures. The first is for user account information, the second is a user modal data structure, and the third is a user validation data structure.

User Account Data Structures

The data structures described here represent levels of detail available for the functions that define or return individual user accounts. The NetUserAdd, NetUserEnum, NetUserGetInfo, and NetUserSetInfo functions use *user_info_0*, *user_info_1*, *user_info_2*, *user_info_10* or *user_info_11*, depending on whether their *level* parameter is 0, 1, 2, 10 or 11.

NetUserGetGroups returns a very simple *group_info_0* data structure, which is described following the syntax section of the function.

User Account Information (Level 0)

```
struct user_info_0 {
  char usri0_name[UNLEN+1];
};
```

where:

- *usri0_name* specifies the name of the user.

User Account Information (Level 1)

```
struct user_info_1 {
  char           usri1_name[UNLEN+1];
  char           usri1_pad_1;
  char           usri1_password[ENCRYPTED_PWLEN];
  long           usri1_password_age;
  unsigned short usri1_priv;
  char far *     usri1_home_dir;
  char far *     usri1_comment;
  unsigned short usri1_flags;
  char far *     usri1_script_path;
};
```

where:

- *usri1_name* specifies name of the user.
- *usri1_pad_1* is for WORD-alignment of fields in the data structure.
- *usri1_password* is the password of a *usri1_name*. The NetUserEnum and NetUserGetInfo functions return a string of spaces to maintain password security. The string can be NULL.
- *usri1_password_age* indicates the number of seconds that have passed since *usri1_password* last changed.

- *usri1_priv* is one of three values indicating the level of privilege assigned *usri1_name*. The ACCESS.H include file defines these values as follows:

Manifest	Value	Privilege
USER_PRIV_GUEST	0	Guest
USER_PRIV_USER	1	User
USER_PRIV_ADMIN	2	Administrator

- *usri1_home_dir* points to an ASCIIZ string containing the path name of the *user_name*'s home directory. The string can be NULL.
- *usri1_comment* points to an optional ASCIIZ string containing an optional comment or remark about the user. The string can be NULL.
- *usri1_flags* is one of the following values that determine whether or not a logon script is to be executed, and whether the user's account is enabled. *usri1_flags* is defined in ACCESS.H as follows:

Manifest	Bit Mask	Meaning
UF_SCRIPT	0x1	Must be 1. Logon script enabled.
UF_ACCOUNTDISABLE	0x2	If 1, user's account disabled.
UF_HOMEDIR_REQUIRED	0x8	If 1, home directory required.
UF_PASSWD_NOTREQD	0x20	If 1, password not required.
UF_PASSWD_CANT_CHANGE	0x40	If 1, user cannot change password.

- *usri1_script_path* points to an ASCIIZ string indicating the path name of the user's logon script (*.cmd* or *.pro* file). The string can be NULL.

User Account Information (Level 2): Information level 2 is needed to accomodate additional information for the UAS database. It will be an extension of *user_info_1* as shown in the following example:

```
struct user_info_2 {
   char                usri2_name[UNLEN+1];
   char                usri2_pad_1;
   char                usri2_password[ENCRYPTED_PWLEN];
   long                usri2_password_age;
   unsigned short      usri2_priv;
   char far *          usri2_home_dir;
   char far *          usri2_comment;
   unsigned short      usri2_flags;
   char far *          usri2_script_path;
   unsigned long       usri2_auth_flags;
   char far *          usri2_full_name;
   char far *          usri2_usr_comment;
   char far *          usri2_parms;
   char far *          usri2_workstations;
   long                usri2_last_logon;
   long                usri2_last_logoff;
   long                usri2_acct_expires;
   unsigned long       usri2_max_storage;
   unsigned short      usri2_units_per_week;
   unsigned char far * usri2_logon_hours;
   unsigned short      usri2_bad_pw_count;
   unsigned short      usri2_num_logons;
   char far *          usri2_logon_server;
   unsigned short      usri2_country_code;
   unsigned short      usri2_code_page;
};
```

where:

- *usri2_name* specifies name of the user.
- *usri2_pad_1* is for WORD-alignment of fields in the data structure.
- *usri2_password* is the password of a *usri1_name*. The NetUserEnum and NetUserGetInfo functions return a string of spaces to maintain password security. The string can be NULL.
- *usri2_password_age* indicates the number of seconds that have passed since *usri1_password* last changed.
- *usri2_priv* is one of three values indicating the level of privilege assigned *usri1_name*. The ACCESS.H include file defines these values as follows:

Manifest	Value	Privilege
USER_PRIV_GUEST	0	Guest
USER_PRIV_USER	1	User
USER_PRIV_ADMIN	2	Administrator

- *usri2_home_dir* points to an ASCIIZ string containing the path name of the *user_name*'s home directory. The string can be NULL.
- *usri2_comment* points to an optional ASCIIZ string containing an optional comment or remark about the user. The string can be NULL.

- *usri2_flags* is one of the following values that determine whether or not a logon script is to be executed, and whether the user's account is enabled. *usri2_flags* is defined in ACCESS.H as follows:

Manifest	Bit Mask	Meaning
UF_SCRIPT	0x1	Must be 1. Logon script enabled.
UF_ACCOUNTDISABLE	0x2	If 1, user's account disabled.
UF_HOMEDIR_REQUIRED	0x8	If 1, home directory required.
UF_PASSWD_NOTREQD	0x20	If 1, password not required.
UF_PASSWD_CANT_CHANGE	0x40	If 1, user cannot change password.

- *usri2_script_path* points to an ASCIIZ string indicating the path name of the user's logon script (*.cmd* or *.pro* file). The string can be NULL.
- *auth_flags* is reserved.
- *full_name* points to an ASCIIZ string containing the full name of the user. The string can be NULL.
- *usr_comment* points to an ASCIIZ string which is a user-settable field. The string can be NULL.
- *parms* points to an ASCIIZ string containing the name of the user's home directory. The string can be NULL.
- *workstations* is a list of requesters from which a user is permitted to log on. A NULL string means all requesters are allowed. (To disallow logon, the account disabled flag must be set.) Up to 8 requesters may be specified. The list of requesters can include IBM NETBIOS permanent names, which are listed as machine IDs, consisting of 12 hexidecimal digits. IBM NETBIOS permanent names are entered in the requester as shown as follows:

  ```
  16DF.02AC.7DE9
  ```

- *last_logon* is the time and date (seconds since 1/1/70) when the last logon occurred. Zero means unknown. This field can be set only by the system.
- *last_logoff* is the time and date (seconds since 1/1/70) when the last logoff occurred. Zero means unknown. This field can be set only by the system.
- *acct_expires* is the time and date (seconds since 1/1/70) when the account will expire. An expired account is the equivalent of a disabled account. An entry of 0xFFFFFFFF means that the account will never expire.
- *max_storage* is a maximum storage allotted for the home directory. Units are K bytes. An entry of 0xFFFFFFFF means that the account will never expire.
- *units_per_week* is the number of equal-length time units into which the week is divided. This value is used to compute the length of the bit string in *logon_hours*. It must be UNITS_PER_WEEK (168) for this release. This field is ignored by NetUserAdd and NetUserSetInfo calls.

- *logon_hours* points to a 21-byte (168 bits) map, each bit representing a unique hour in a week. The first bit (bit 0, word 0) is Sunday, 0:00 to 0:59. Bit 1, word 0 is Sunday, 1:00 to 1:59, and so on. If a bit is set in this bitmap, it means that logon is allowed. If a bit is cleared, it means logon is not allowed.

  Note: A NULL pointer permits access at all times.

- *bad_pw_count* is the number of attempts to validate a bad password. A value of -1 means unknown. This field is ignored by NetUserAdd and NetUserSetInfo calls.

- *num_logons* is the number of instances of logons to the account. A value of -1 means unknown. This field is ignored by NetUserAdd and NetUserSetInfo calls.

- *logon_server* points to a NULL string.

- *country_code* is the OS/2 country code for the user's language choice. This is used by the OS/2 LAN Requester/Server software to generate messages in the appropriate language whenever possible.

- *code_page* is the OS/2 code page for the language choice of the user.

User Account Information (Level 10): The following structure is provided to allow users to retrieve limited information about themselves.

```
struct user_info_10 {
   char           usri10_name[UNLEN+1];
   char           usri10_pad_1;
   char far *     usri10_comment;
   char far *     usri10_usr_comment;
   char far *     usri10_full_name;
};
```

where:

- *usri10_name* is the name of the user.

- *usri10_pad_1* is for WORD-alignment of fields in the data structure.

- *usri10_comment* points to an ASCIIZ string which is for remarks or comments and can contain any type of text. The string can be NULL.

- *usr_comment* points to an ASCIIZ string which a user-settable field. The string can be NULL.

- *usri10_full_name* is the full name of user.

User Account Information (Level 11): The following structure is provided to allow users to retrieve more information about other users.

```
struct user_info_11 {
   char                usri11_name[UNLEN+1];
   char                usri11_pad_1;
   char far *          usri1_comment;
   char far *          usri11_usr_comment;
   char far *          usri11_full_name;
   unsigned short      usri11_priv;
   unsigned long       usri11_auth_flags;
   long                usri11_password_age;
   char far *          usri11_home_dir;
   char far *          usri11_parms;
   long                usri11_last_logon;
   long                usri11_last_logoff;
   long                usri11_bad_pw_count;
   unsigned short      usri11_num_logons;
   char far *          usri11_logon_server;
   unsigned short      usri11_country_code;
   char far *          usri11_workstations;
   unsigned long       usri11_max_storage;
   unsigned short      usri11_units_per_week;
   unsigned char far * usri11_logon_hours;
   unsigned short      usri11_code_page;
};
```

where:

- *usri11_name* is the name of the user.
- *usri11_pad_1* is for WORD-alignment of fields in the data structure.
- *usri11_comment* is for remarks or comment; it can contain any type of text. The string can be NULL.
- *usr_comment* points to an ASCIIZ string which is a user-settable comment field. The string can be NULL.
- *usri11_full_name* points to an ASCIIZ string containing the full name of the user. The string can be NULL.
- *usri11_priv* is the user's privilege level. It can be one of the following:

Manifest	Value	Privilege
USER_PRIV_GUEST	0	Guest
USER_PRIV_USER	1	User
USER_PRIV_ADMIN	2	Administrator

- *usri11_auth_flags* is reserved.
- *usri11_password_age* is the time (in seconds) since the password was last changed.
- *usri11_home_dir* is the user's home directory. It can be a null string. The absolute path can be local or UNC.
- *usri11_parms* points to an ASCIIZ string containing the name of the user's home directory. The string can be NULL.

- *usri11_last_logon* is the time and date (seconds since 1/1/70) when the last logon occurred. A value of zero means unknown.
- *usri11_last_logoff* is the time and date (seconds since 1/1/70) when the last logoff occurred. A value of zero means unknown.
- *usri11_bad_pw_count* is the number of attempts to validate a bad password. A value of -1 means unknown.
- *usri11_num_logons* is the number of instances of logons to the account. A value of -1 means unknown.
- *usri11_logon_server* is the computer to handle logon requests for a user.
- *usri11_country_code* is the OS/2 country code for the user's language choice. This is used by the OS/2 LAN Requester/Server software to generate messages in the appropriate language whenever possible.
- *usri11_workstations* is a list of requesters from which a user is permitted to log on. A NULL string means all requesters are allowed. (To disallow logon, the account disabled flag must be set.) Up to 8 requesters may be specified. The list of requesters can include IBM NETBIOS permanent names, which are listed as machine IDs, consisting of 12 hexidecimal digits. IBM NETBIOS permanent names are entered in the requester as shown as follows:

  ```
  16DF.02AC.7DE9
  ```

- *usri11_max_storage* is a maximum storage allotted for the home directory. Units are K bytes. An entry of 0xFFFFFFFF means that the account will never expire.
- *usri11_units_per_week* is the number of equal-length time units into which the week is divided. This value is used to compute the length of the bit string in *logon_hours*. It must be UNITS_PER_WEEK (168) for this release.
- *usri11_logon_hours* is a 21-byte (168 bits) map, each bit representing a unique hour in a week. The first bit (bit 0, word 0) is Sunday, 0:00 - 0:59. Bit 1, word 0 is Sunday, 1:00 - 1:59, and so on. If a bit is set in this bitmap, it means that logon is allowed. If a bit is cleared, it means logon is not allowed.
- *usri11_code_page* is the OS/2 code page for the language choice of the user.

  Note: A NULL entry permits access at all times.

User Modals Data Structure

In order to control global modals, the following data structure is used.

User Modals Information (Level 0)

```
struct  user_modals_info_0
{
  unsigned short  usrmod0_min_passwd_len;
  unsigned long   usrmod0_max_passwd_age;
  unsigned long   usrmod0_min_passwd_age;
  unsigned long   usrmod0_force_logoff;
  unsigned short  usrmod0_password_hist_len;
  unsigned short  usrmod0_max_reserved1
};
```

where:

- *min_passwd_len* is the minimum password length. The range of values is 0 to MAX_PASSWD_LEN.
- *max_passwd_age* is the maximum time (in seconds) since the password was last changed, and for which the current password is valid. A value of 0xFFFFFFFF (TIMEQ_FOREVER) allows the password to be valid forever. The minimum value is one day.
- *min_passwd_age* is the minimum time (in seconds) since the password was last changed, before which the current password is allowed to be changed. A value of 0 means there is no delay required between password updates.
- *force_logoff* is the length of time (in seconds) after the valid logon hours that the user should be forced off the network. The user will never be forced off if the value is TIMEQ_FOREVER (0xFFFFFFFF), or be forced off immediately if the value is 0. Any value between these can also be used.
- *password_hist_len* is the length of the password history, that is, the number of passwords in the history buffer that are scanned versus the new password in a NetUserPasswordSet attempt. The new password may not match any of the entries scanned. The history is of ENCRYPTED passwords. It can be 0 to DEF_MAX_PWHIST (currently 8).
- *max_reserved1* is not used.

User Modals Information (Level 1)

```
struct  user_modals_info_1
{
  unsigned short  usrmod1_role;
  char far *      usrmod1_primary;
};
```

where:

- *role* is the role of this database under a single system image (SSI). It can be one of the following:

Manifest	Value	Meaning
UAS_ROLE_STANDALONE	0	Standalone database
UAS_ROLE_MEMBER	1	Member database in the domain
UAS_ROLE_BACKUP	2	Backup database in the domain
UAS_ROLE_PRIMARY	3	Primary database in the domain

  Without SSI, this field should always be set to STANDALONE.

- *primary* is the name of the primary domain to which this database belongs. It should match the primary domain name of the requester software on the local machine.

User Validation Data Structures

User Validation Information (Requester)

```
struct user_logon_req_1 {
   char           usrreq1_name[UNLEN+1];
   char           usrreq1_pad_1;
   char           usrreq1_password[SESSION_PWLEN];
   char far *     usrreq1_workstation;
};
```

where:

- *usrreq1_name* is the account name to which the user is attempting to log on.
- *usrreq1_pad* is for WORD-alignment of data fields.
- *usrreq1_password* is the plain text ASCIIZ password.
- *usrreq1_workstation* is the ASCIIZ string representing the requester from which the user is logging on. Can be NULL for local; ((long) -1) for unknown.

User Validation Information (Level 0): The following is the logon data structure:

```
struct user_logon_info_0 {
   char           usrlog0_eff_name[UNLEN+1];
   char           usrlog0_pad_1;
};
```

where:

- *usrlog0_eff_name* is the name of the account to which the user was logged on.
- *usrlog0_pad* is for WORD-alignment of the data structure.

User Validation Information (Level 1)

```
struct user_logon_info_1 {
   unsigned short usrlog1_code;
   char           usrlog1_eff_name[UNLEN+1];
   char           usrlog1_pad_1;
   unsigned short usrlog1_priv;
   unsigned long  usrlog1_auth_flags;
   unsigned short usrlog1_num_logons;
   unsigned short usrlog1_bad_pw_count;
   long           usrlog1_last_logon;
   unsigned long  usrlog1_last_logoff;
   unsigned long  usrlog1_logoff_time;
   unsigned long  usrlog1_kickoff_time;
   long           usrlog1_password_age;
   unsigned long  usrlog1_pw_can_change;
   unsigned long  usrlog1_pw_must_change;
   char far *     usrlog1_computer;
   char far *     usrlog1_domain;
   char far *     usrlog1_script_path;
   unsigned long  usrlog1_reserved1;
};
```

where:

- *usrlog1_code* is the code explaining an action taken. The following are possible values:

NERR_Success	No problems were encountered.
NERR_PasswordExpired	The user has an account, but the user's password has expired. No other conditions of logon have been checked.
NERR_InvalidWorkstation	The user was attempting to log on from an invalid requester.
NERR_InvalidLogonHours	The user was attempting to log on at an invalid time.
ERROR_ACCESS_DENIED	Some condition of logon has not been met.
NERR_StandaloneLogon	No domain controller could be found to validate the user. Script processing was not performed.
NERR_NonValidatedLogon	The logon request was serviced by a 1.X logon server. Logon script processing was performed.
NERR_LogonScriptError	An error occurred processing logon script.

  The *user_logon_info_1* data structure is returned by both the NetUserValidate2 API and the NetWkstaSetUID2 API. The following codes may be returned by NetwkstaSetUID2, but are never returned by NetUserValidate2.

NERR_StandaloneLogon	No DC could be found to validate the user.
NERR_LogonScriptError	An error occurred processing the logon script.

  The other fields in the *user_logon_info_1* data structure are valid only when the API returns 0, and the code is VALIDATED_LOGON.

- *usrlog1_eff_name* is the name of the account to which the user was logged on.
- *usrlog1_pad_1* is for WORD-alignment of data fields.
- *usrlog1_priv* is the user's privilege level. It can be one of the following:

Manifest	Value	Privilege
USER_PRIV_GUEST	0	Guest
USER_PRIV_USER	1	User
USER_PRIV_ADMIN	2	Administrator

- *usrlog1_auth_flags* is reserved.
- *usrlog1_num_logons* is the number of instances of logons to the account. A value of -1 means unknown.
- *usrlog1_bad_pw_count* is the number of attempts to validate a bad password. A value of -1 means unknown.
- *usrlog1_last_logon* is the time and date (seconds since 1/1/70) when the last logon occurred. A value of zero means unknown.
- *usrlog1_last_logoff* is the time and date (seconds since 1/1/70) when the last logoff occurred. A value of zero means unknown.
- *usrlog1_logoff_time* is the time (in seconds) and date (since 1/1/70) when the user should log off. A value of ((long) -1) means never.
- *usrlog1_kickoff_time* is the time (in seconds) when the system (server, communications manager, and so on.) should force the user to log off. This is

the sum of the logoff time and the grace period. A value of ((long) -1) means never.

- *usrlog1_password_age* is the time (in seconds) since the password was last changed.
- *usrlog1_pw_can_change* is the time (in seconds) and date (since 1/1/70) when the user can change his password. A value of -1 prevents users from changing their passwords.
- *usrlog1_pw_must_change* is the time (in seconds) and the date (since 1/1/70) when the password must be changed.
- *usrlog1_computer* is the domain logged on by. It can be 0 for local.
- *usrlog1_domain* is the domain logged on by. It can be 0 for local.
- *usrlog1_script_path* is the relative path to the logon script of an account.
- *usrlog1_reserved1* is reserved for internal use only. Must be zero.

User Validation Information (Level 2)

```
struct user_logon_info_2 {
  char          usrlog2_eff_name[UNLEN+1];
  char          usrlog2_pad_1;
  char *        usrlog2_computer;
  char *        usrlog2_fullname;
  char *        usrlog2_usrcomment;
  unsigned long  usrlog2_logon_time;
}
```

where:

- *usrlog2_eff_name* is the name of the account to which the user was logged on.
- *usrlog2_pad_1* is for WORD-alignment of data fields.
- *usrlog2_computer* is the server logged on. Can be zero for local.
- *usrlog2_fullname* is the full name of the user.
- *usrlog2_comment* points to an ASCIIZ string which is a user-settable field for user comments. The string can be NULL.
- *usrlog2_logon_time* is the time and date (seconds since 1/1/70) when the user logged on. ((long) -1 means not available.)

User Validation Information (Logoff): The following is the logoff data structure:

```
struct user_logoff_info_1 {
   unsigned short usrlog1_code;
   unsigned long  usrlogf1_duration;
   unsigned short usrlogf1_num_logons;
};
```

where:

- *usrlogf1_code* is dependent on the calling API. It is specified separately for each API.
- *usrlogf1_duration* is the duration of a logon. Can be 0 for unknown.

- *usrlogf1_num_logons* is the number of logons by a user name. Can be -1 for unknown.

Related Information

For information on:

- Logon scripts—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- Share resource access permissions—See "Share Category" on page 3-337.

NetUserAdd

The NetUserAdd (admin, DOS) function establishes an account for a user in the use accounts subsystem (UAS) database and assigns a password and privilege level.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserAdd(servername, level, buf, buflen)
char far *      servername;
short           level;
char far *      buf;
unsigned short  buflen;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (1 or 2) provided by the *user_info_1* or *user_info_2* data structures.
- *buf* points to the *user_info_1* or *user_info_2* data structures.
- *buflen* specifies the size (in bytes) of the *buf* memory area.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_BadUsername	2202	The user name or group name parameter is invalid.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_GroupExists	2223	The group name is already in use.
NERR_UserExists	2224	The user account already exists.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFNoRoom	2228	There are too many names in the access control file.

Manifest	Value	Meaning
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_PasswordTooShort	2245	The password is shorter than required.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_BadAsgType	2251	This asg_type is invalid.
NERR_DeviceIsShared	2252	This device is already being shared.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosDevIOCtl(IOC_ENCRYPT)
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NIOC_ENCRYPT)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize [-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite.

Remarks

For add calls at level 1, the additional fields in the *user_info_2* data structures are initialized to appropriate default values. These can be set to the desired values by subsequent NetUserSetInfo calls. The default values are as follows:

usri2_full_name	*useri2_name*
usri2_usr_comment	None (NULL string)
usri2_parms	None (NULL string)

usri2_workstations	All (NULL string)
usri2_acct_expires	Never (0xFFFFFFFF)
usri2_max_storage	Unlimited (0xFFFFFFFF)
usri2_logon_hours[]	Logon allowed anytime (each element 0xFF; all bits set to 1)
usri2_logon_server	Domain controller (NULL string)
usri2_country_code	Current country code on the server
usri2_code_page	Current code page on the server
usri2_auth_flags	None (0).

The following fields must be set to 0:

usri2_password_age	Set so that base time equals time of creation
reserved	All reserved fields.

Related Information

For information on:

- Listing user accounts on a server—See "NetUserEnum" on page 3-404.
- Logon scripts—See the *IBM Operating System/2 Local Area Network Server Version 1.2 Network Administrator's Guide.*
- Modifying a user account—See "NetUserSetInfo" on page 3-426.
- Modifying a user's group memberships—See "Group Category" on page 3-110.
- Removing a user account—See "NetUserDel" on page 3-401.

NetUserDel

The NetUserDel (admin, DOS) function removes an account from the user account subsystem database, ending all access to the resources in the system.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserDel(servername, username)
char far *  servername;
char far *  username;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *username* points to an ASCIIZ string containing the user name to be deleted.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.

Manifest	Value	Meaning
NERR_ServerNotStarted	2114	The Server service has not been started.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_LastAdmin	2452	The last administrator cannot be deleted.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NIOC_ENCRYPT)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize [-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite.

Remarks

Only a user or application with administrative privileges can delete another user's account.

Deleting an account also deletes all references to that account in all resource permission access records and from groups in the system.

If the account being deleted is the last account in the database with administrative privilege, the error NERR_LastAdmin is returned and the delete function fails.

Related Information

For information on:

- Creating a user account of use of server resources—See "NetUserAdd" on page 3-397.
- Listing user accounts on a server—See "NetUserEnum" on page 3-404.
- Modifying a user account—See "NetUserSetInfo" on page 3-426.

NetUserEnum

The NetUserEnum (partially admin, DOS) function returns information about all accounts on a server.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserEnum(servername, level, buf, buflen,
            entriesread, totalentries)
char far *            servername;
short                 level;
char far *            buf;
unsigned short        buflen;
unsigned short far *  entriesread;
unsigned short far *  totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (0, 1, 2 or 10) returned in the *user_info* data structure.
- *buf* points to the *user_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_BadAsgType	2251	This asg_type is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)

- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosRead
- DosSemClear.

Remarks

A user without administrative privileges can call this API with levels 0 or 10 only.

NetUserEnum returns *user_info_0*, *user_info_1*, *user_info_2*, or *user_info_10*, components. If *level* is 1 or 2, the password component of each data structure will contain a string of spaces to maintain password security.

Related Information

For information on:

- Listing all groups to which a user belongs—See "NetUserGetGroups" on page 3-407.
- Retrieving the status of a particular user account—See "NetUserGetInfo" on page 3-410.

NetUserGetGroups

The NetUserGetGroups (partially admin, DOS) function lists the names of all groups in the user accounts subsystem (UAS) database to which a particular user belongs. Users can use the NetUserGetGroups function on their user names.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserGetGroups(servername, username, level, buf, buflen,
                 entriesread, totalentries)
char far *           servername;
char far *           username;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far * entriesread;
unsigned short far * totalentries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *username* points to an ASCIIZ string indicating the name of the user to search for in each group account.
- *level* specifies the level of detail (0) returned in the *group_info_0* data structure (described below).
- *buf* points to the *group_info_0* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entriesread* points to an unsigned short integer indicating the number of entries that were returned to *buf*.
- *totalentries* points to an unsigned short integer indicating the number of entries that were available.

Group Membership (Level 0)

```
struct group_info_0 {
 char grpi0_name[GNLEN+1];
};
```

where:

- *grpi0_name* contains the names of the groups to which a user belongs.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.

Manifest	Value	Meaning
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_GroupExists	2223	The group name is already in use.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.

Manifest	Value	Meaning
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize [-ERROR_DISK_FULL]
- DosRead
- DosSemClear.

Remarks

NetUserGetGroups returns an array of *group_info_0* data structures, indicating the names of all groups to which the user belongs.

This is functionally equivalent to an Enum call because it enumerates the groups of which a user is a member.

Related Information

For information on:

- Retrieving the status of a user account—See "NetUserGetInfo" on page 3-410.
- Retrieving and setting information about groups and their members on a server—See "Group Category" on page 3-110.

NetUserGetInfo

The NetUserGetInfo (partially admin, DOS) function retrieves information about a particular account on a server.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserGetInfo(servername, username, level,
               buf, buflen, totalavail)
char far *           servername;
char far *           username;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far * totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *username* points to an ASCIIZ string indicating the user's account information to retrieve.
- *level* specifies the level of detail (0, 1, 2, 10, 11) returned in the *user_info* data structure.
- *buf* points to the *user_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to an unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize [-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite.

Remarks

NetUserGetInfo can be used to retrieve information on the user's privileges, home directory, and the status of an account. When *level* is 1 or 2, the password component of the *user_info* data structure contains a string of spaces to maintain password security.

A user without administrative privileges can call this API with levels 0, 10, and 11 only. Level 11 is available only for the account to which the user is logged on.

Related Information

For information on:

- Modifying a user's account—See "NetUserSetInfo" on page 3-426.
- Retrieving a list of groups to which a user belongs—See "NetUserGetGroups" on page 3-407.
- Retrieving information on all user accounts on a server—See "NetUserEnum" on page 3-404.

NetUserModalsGet

The NetUserModalsGet (DOS) function gets global modals-related information for all users and groups in the user accounts subsystem (UAS) database.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserModalsGet( servername, level, buf, buflen, totalavail)

char far *           servername;
short                level;
char far *           buf;
unsigned short       buflen;
unsigned short far *  totalavail;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (0 or 1) supplied to the *user_modals_info* data structure.
- *buf* points the *user_modals_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *totalavail* points to the unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidComputer	2351	The specified computer name is invalid.

Other error return codes may be returned from the following OS/2 functions:

- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]

- DosRead
- DosSemClear.

Remarks

This function can be called even when the UAS system is not active, provided a valid UAS database exists.

A user without administrative privileges can call this API with level 0 only.

NetUserModalsSet

The NetUserModalsSet (admin, DOS) function sets global modals-related information for all users and groups in the user accounts subsystem (UAS) database.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserModalsSet( servername, level, buf, buflen, parmnum)

char far *          servername;
short               level;
char far *          buf;
unsigned short      buflen;
short               parmnum;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *level* specifies the level of detail (0 or 1) supplied to the *user_modals_info* data structure.
- *buf* points to the data structure if *parmnum* is zero. Otherwise, *buf* points to the specific data component that will be changed.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* determines which parameter to set. If *parmnum* is zero, *level* must be 0 and *buff* must contain the *user_modals_info_0* data structure.

All fields in the *user_modals_info_0* data structure, except *accsys_status*, are settable.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_InvalidUASOp	2451	This operation is not permitted when the Netlogon service is running.

Other error return codes may be returned from the following OS/2 functions:

- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosRead
- DosSemClear
- DosWrite.

Remarks

This function can be called even when the UAS system is not active, provided a valid UAS database exists. SetInfo with level 1 can be called only when the accounts subsystem is inactive.

The accounts system *must* be inactive in order to change the name of the database domain.

NETLOGON *cannot* be running when setting the *role* field.

NetUserPasswordSet

The NetUserPasswordSet (DOS) function changes the password stored in a user's account in the system.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserPasswordSet(servername, username, oldpasswd, newpasswd)
char far *  servername;
char far *  username;
char far *  oldpasswd;
char far *  newpasswd;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *username* points to an ASCIIZ string containing the name of the user whose password will be changed.
- *oldpasswd* points to an ASCIIZ string specifying the user's current password.
- *newpasswd* points to an ASCIIZ string specifying a new password for the user.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PASSWORD	86	The specified password is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_BadPassword	2203	The password parameter is invalid.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_AccountExpired	2239	The user account has expired.
NERR_PasswordCantChange	2243	This password cannot change.
NERR_PasswordHistConflict	2244	This password cannot be used now.
NERR_PasswordTooShort	2245	The password is shorter than required.
NERR_PasswordTooRecent	2246	The password is too recent to change.

Manifest	Value	Meaning
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosDevIOCtl(IOC_ENCRYPT)
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NIOC_ENCRYPT)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize [-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite
- RdrDESEncrypt(NIOC_ENCRYPT) (pseudo FsCtl).

Remarks

Passwords are case-sensitive.

The password will be set to *newpasswd* only if the *oldpasswd* matches the current password for the user and the *newpasswd* is not the same as the *usrmodal10_phistlen* password. This call allows users to change their own password without administrative privilege. Password update restrictions apply.

If users have administrative privilege and need to change a user's password, without knowing their old password, they can use NetUserSetInfo instead.

NetUserPasswordSet, if successful to the domain controller of a single system image (SSI) domain to which the user is currently logged on, also sets the current "default" password of the requester. This allows the user to connect to all servers in the domain, since the new password should rapidly replicate throughtout the domain.

Special Server Actions

When used remotely, this API is handled in a special way. It must be allowed to succeed even when the user's current password is expired. If the password of the account has expired but the account could otherwise log on, the server permits a NetUserPasswordSet to succeed.

The server does this by examining the session setup, and if the session setup is followed by a PasswordSet transaction, the server will validate the session setup, process the transaction, and then close the session.

Related Information

For information on setting a password, see "NetUserSetInfo" on page 3-426.

NetUserSetGroups

The NetUserSetGroups (admin, DOS) function sets the groups of which a user is a member.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserSetGroups( servername, username, level, buf, buflen, entries)

char far *          servername;
char far *          username;
short               level;
char far *          buf;
unsigned short      buflen;
unsigned short      entries;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *username* points to an ASCIIZ string containing the name of the user whose group is being set.
- *level* specifies the level of detail (0) supplied to the *group_info_0* data structure.
- *buf* points the *group_info* data structure.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *entries* points to the unsigned short integer indicating the number of entries that were supplied in the buffer.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_GroupNotFound	2220	The group does not exist.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_SpeGroupOp	2234	This operation is not allowed on this special group.

Manifest	Value	Meaning
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize [-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite.

NetUserSetInfo

The NetUserSetInfo (partially admin, DOS) function modifies a user's account in the system.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserSetInfo(servername, username, level,
               buf, buflen, parmnum)
char far *     servername;
char far *     username;
short          level;
char far *     buf;
unsigned short buflen;
short          parmnum;
```

where:

- *servername* points to an ASCIIZ string containing the name of the remote server on which the function is to execute. A NULL pointer or string specifies a local server.
- *username* points to an ASCIIZ string specifying which user's account to modify.
- *level* specifies the level of detail (1 or 2) provided by the *user_info* data structure.
- *buf* points to the data structure if *parmnum* is zero. Otherwise, *buf* points to the specific data component that will be changed.

Settable Fields	Admin	Owner
name	x	
password	x	
priv	x	
home_dir	x	
comment	x	
flags	x	
script_path	x	
auth_flags	x	
full_name	x	
usr_comment	x	x
parms	x	x
workstations	x	
acct_expires	x	
max_storage	x	
logon_hours	x	
logon_server	x	

Settable Fields	Admin	Owner
country_code	x	x
code_page	x	x

- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *parmnum* determines whether *buf* contains a complete *user_info* data structure or a single component. If *parmnum* is 0, *level* can be 1 or 2, and *buf* must contain a complete *user_info_1* or *user_info_2* data structure. Otherwise, *parmnum* must specify the ordinal position value for one of the following data structure components, as defined in ACCESS.H, as follows:

Manifest	Value	Component
PARMNUM_NAME	1	*usrix_name*
PARMNUM_PASSWD	3	*usrix_password*
PARMNUM_PRIV	5	*usrix_priv*
PARMNUM_DIR	6	*usrix_home_dir*
PARMNUM_COMMENT	7	*usrix_comment*
PARMNUM_USER_FLAGS	8	*usrix_user_flags*
PARMNUM_SCRIPT_PATH	9	*usrix_script_path*
PARMNUM_AUTH_FLAGS	25	*usri2_auth_flags*
PARMNUM_FULL_NAME	10	*usri2_full_name*
PARMNUM_USR_COMMENT	11	*usri2_usr_comment*
PARMNUM_PARMS	12	*usri2_parms*
PARMNUM_WORKSTATIONS	13	*usri2_workstations*
PARMNUM_ACCT_EXPIRES	16	*usri2_acct_expires*
PARMNUM_MAX_STORAGE	17	*usri2_max_storage*
PARMNUM_UNITS_PER_WEEK	24	*usri2_units_per_week*
PARMNUM_LOGON_HOURS	18	*usri2_logon_hours*
PARMNUM_LOGON_SERVER	21	*usri2_logon_server*
PARMNUM_COUNTRY_CODE	22	*usri2_country_code*
PARMNUM_CODE_PAGE	25	*usri2_code_page*

Note: x = 1 or 2.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.

Manifest	Value	Meaning
ERROR_NOT_ENOUGH_MEMORY	8	Sufficient memory is not available.
ERROR_SEEK	25	The seek is invalid.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.
NERR_NetNotStarted	2102	The redirector NETWKSTA.EXE has not been started.
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_RemoteOnly	2106	This operation is not supported on workstations.
NERR_BufTooSmall	2123	The buffer is too small for fixed-length data.
NERR_OS2IoctlError	2134	An internal error occurred when calling the workstation driver.
NERR_WkstaNotStarted	2138	The Requester service has not been started.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_InternalError	2140	An internal error has occurred.
NERR_BadTransactConfig	2141	The server is not configured for transactions.
NERR_InvalidAPI	2142	The requested API is not supported on the remote server.
NERR_ACFNotFound	2219	The Net.ACC file is missing.
NERR_UserNotFound	2221	The user name cannot be found.

Manifest	Value	Meaning
NERR_NotPrimary	2226	The UAS database is replicant and will not allow updates.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_PasswordTooShort	2245	The password is shorter than required.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_BadAsgType	2251	This asg_type is invalid.
NERR_DeviceIsShared	2252	This device is already being shared.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_LastAdmin	2452	The last administrator cannot be deleted.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosDevIOCtl(IOC_ENCRYPT)
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NETTRANSACTION)
- DosFsCrl(NIOC_ENCRYPT)
- DosFsCtl(NULLTRANSACT)
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize [-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite
- RdrDESEncrypt (NIOC_ENCRYPT) (pseudo FsCtl).

Remarks

NetUserSetInfo can be called to change a user's password only by users or applications having administrative privileges. Also note that the only restriction enforced when called by an administrator is that the new password length must be consistent with system modals.

Note that all settable fields can be set by a user with administrative privilege. Any user may set the fields marked under "owner" for his own account. To do this, the user must use the *parmnum* option and cannot pass the whole structure. NULL_USERSETINFO_PASSWD must be used with *parmnum* 0 option if it is desired that the password not be changed.

If this function is to change the privilege of or disable the last account in the database with administrative privilege, the error NERR_LastAdmin is returned and the function fails.

Related Information

For information on a particular user name on a server, see "NetUserGetInfo" on page 3-410.

NetUserValidate2

The NetUserValidate2 (local) function validates a user with its password. It checks if the user can log on based on logon restrictions defined for the account.

Syntax

```
#include <netcons.h>
#include <access.h>

unsigned far pascal
NetUserValidate2( reserved1, level, buf, buflen, reserved2, totalavail )

char far             reserved1;
short                level
char far *           buf
unsigned short       buflen;
unsigned short       reserved2;
unsigned short far *  totalavail;
```

where:

- *reserved1* must be NULL.
- *level* specifies the level of detail (1) supplied to the *user_logon* data structure.
- *buf* points to the *user_logon* data structure.

  Buffer contents on call: A structure *user_logon_req_1*. The *usrreq1_password* is a plain text password.

  Buffer contents on response: A structure *user_logon_info_1*.
- *buflen* specifies the size (in bytes) of the *buf* memory area.
- *reserved2* is reserved and must be zero.
- *totalavail* points to the unsigned short integer indicating the number of bytes of information that were available.

Return Codes

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
ERROR_SEEK	25	The seek is invalid.
ERROR_NOT_SUPPORTED	50	This request is not supported by the network.
ERROR_INVALID_PARAMETER	87	The specified parameter is invalid.
ERROR_INVALID_LEVEL	124	The Level parameter is invalid.
ERROR_MORE_DATA	234	Additional data is available, but the buffer is too small.

Manifest	Value	Meaning
NERR_ShareMem	2104	An internal error occurred—the network cannot access a shared memory segment.
NERR_BrowserNotStarted	2139	The requested information is not available.
NERR_UserNotFound	2221	The user name cannot be found.
NERR_GroupExists	2223	The group name is already in use.
NERR_ACFNotLoaded	2227	The UAS database has not been started.
NERR_ACFFileIOFail	2229	An error was encountered in accessing the accounts database.
NERR_InvalidDatabase	2247	The UAS database file is corrupted.
NERR_InvalidComputer	2351	The specified computer name is invalid.
NERR_CanNotGrowUASFile	2456	It is not possible to grow the UAS file.

Other error return codes may be returned from the following OS/2 functions:

- DosAllocSeg
- DosChgFilePtr
- DosFsRamSemClear
- DosFreeSeg
- DosFsCtl(NetGetRdrAddr)
- DosGetShrSeg[-ERROR_FILE_NOT_FOUND]
- DosNewSize [-ERROR_DISK_FULL]
- DosQFileInfo
- DosRead
- DosSemClear
- DosWrite
- RdrDESEncrypt(NIOC_ENCRYPT) (pseudo FsCtl).

Remarks

If the account named in *usrreq1_name* has no password, then any value in the password field will match.

The behavior of this API is undefined if the value of *reserved1* is not zero.

The *code* field in the *user_logon_info_1* data structure can have the following error codes:

Manifest	Value	Meaning
NERR_SUCCESS	0	No errors were encountered.
ERROR_ACCESS_DENIED	5	Administrative privilege is required.
NERR_InvalidWorkstation	2240	The user is not allowed to log on from this requester.
NERR_InvalidLogonHours	2241	The user is not allowed to log on at this time.
NERR_PasswordExpired	2242	The password has expired.

The rest of the *user_info_1* structure is valid only when *code* is NERR_Success and the API returned either NERR_Success or ERROR_MORE_DATA.

The bad password count field in the user's record is incremented if the password failed to match. A successful match resets this field.

The behavior of this API is undefined if the value of *reserved1* is not zero.

This function works on the local system only. If there is a standalone UAS database (NET.ACC) existing on the requester, this API, when executed, will validate against the local UAS database instead of the UAS database of the domain.

Appendix A. Include Files

This appendix lists and describes each of the include files provided with the OS/2 LAN Requester/Server applications program interface (API). The include files declare information such as constants, storage classes, and function parameter types that the OS/2 LAN Requester/Server API functions require. Each include file declares information pertinent to a particular category of functions described in Chapter 3, "API Function Descriptions." All OS/2 LAN Requester/Server include files are stored in the \IBMLAN\NETSRC\H directory.

The API functions use the include files listed as follows:

API Category	Include Files	Meaning
All	OS2.H	Declares structures, constants, and functions in the base operating system. **Note:** The OS/2.H include file is not part of the OS/2 LAN Requester/Server include files.
	NETCONS.H	Declares constants used with various components of OS/2 LAN Requester/Server. Used with all OS/2 LAN Requester/Server functions.
	NETERR.H	Declares error manifests for LAN Server return codes. Used with all OS/2 LAN Requester/Server functions.
Access Permission	ACCESS.H	Declares structures, constants, and functions that extract information and control the access privileges for the shared resources of a server. Used with functions in the Access Permissions, Groups, and User categories.
Alert	ALERT.H	Declares message constants used by the alerter service program in conjunction with functions in the Alert category.
	ALERTMSG.H	Declares structures, constants, or functions that wait for specific events to occur, enabling the operation of several network services. Used with functions in the Alert category.

API Category	Include Files	Meaning
Audit	AUDIT.H	Declares structures and constants used for adding events to the audit log of a requester or server. Used with functions in the Auditing category.
Configuration	CONFIG.H	Declares functions that retrieve the parameter of a particular component from the IBMLAN.INI file. Used with functions in the Configuration category.
Connection	SHARES.H	Declares structures, constants, and functions that list information about the resources a server is sharing. Used with functions in the Connection, File, Session, and Share categories.
Domain	ACCESS.H	Declares structures, constants, and functions that extract information and control the access privileges for the shared resources of a server. Used with functions in the Access Permissions, Groups, and User categories.
Error Logging	ERRLOG.H	Declares structures and functions that clear, close, log on to, open, or read the error log file. Also declares the constants that define the error codes related to the operating system or redirector. Used with functions in the Error Logging category.
File	SHARES.H	Declares structures, constants, and functions that list information about the resources a server is sharing. Used with functions in the Connection, File, Session, and Share categories.
Handle	CHARDEV.H	Declares structures, constants, and functions used to modify, list, retrieve, and delete shared serial devices and queues. Used with functions in the Serial Device category.

API Category	Include Files	Meaning
Group	ACCESS.H	Declares structures, constants, and functions that extract information and control the access privileges for the shared resources of a server. Used with functions in the Access Permissions, Groups, and User categories.
Mailslot	MAILSLOT.H	Declares functions that create, delete, read, and write mailslots on a server. Used with functions in the Mailslot category.
Message	MESSAGE.H	Declares structures, constants, and functions that send one-way messages, read messages, or log messages. Used with functions in the Message category.
Remote Utility	REMUTIL.H	Declares data structures, constants, and functions used by remote utilities. Used with functions in the Remote Utility category.
Requester	WKSTA.H	Declares structures, constants, and functions that control the working environment of a requester. Used with functions in the Requester category.
	ACCESS.H	Declares structures, constants, and functions that extract information and control the access privileges for the shared resources of a server. Used with functions in the Access Permissions, Groups, and User categories.
Serial Device	CHARDEV.H	Declares structures, constants, and functions used to modify, list, retrieve, and delete shared serial devices and queues. Used with functions in the Serial Device category.

API Category	Include Files	Meaning
Server	SERVER.H	Declares structures, constants, and functions that retrieve or set information regarding a server. Used with functions in the Server category.
Service	SERVICE.H	Declares structures, constants, and functions that install or control network service programs on a server. Used with functions in the Service category.
Session	SHARES.H	Declares structures, constants, and functions that list information about the resources a server is sharing. Used with functions in the Connection, File, Session, and Share categories.
Share	SHARES.H	Declares structures, constants, and functions that list information about the resources a server is sharing. Used with functions in the Connection, File, Session, and Share categories.
Spooler	PMSPL.H	Part of the OS/2 Presentation Manager™ include files which declare structures, constants, and functions used to print in PM Spooler.
Statistics	NETSTATS.H	Declares structures, constants, and functions that clear or retrieve statistical information about the performance of a server. Used with functions in the Statistics category.
Use	ACCESS.H	Declares structures, constants, and functions that extract information and control the access privileges for the shared resources of a server. Used with functions in the Access Permission, Group, and User categories.

API Category	Include Files	Meaning
User	USER.H	Declares structures, constants, and functions that retrieve or control user information. Used with functions in the User category.

Presentation Manager is a trademark of International Business Machines Corporation.

Appendix B. Function Libraries

This appendix describes the different libraries that a program can be linked to when calling a OS/2 LAN Requester/Server function. Programs link to standard libraries (.LIB) and dynamically linked libraries (.DLL). Standard libraries provide information such as the name of a dynamically linked run-time library to the relocatable object code at link time. Dynamically linked libraries contain the actual assembler code of a function and are executed at run time. In addition, this appendix provides an alphabetical list of all OS/2 LAN Requester/Server functions with each required library, other software required to execute each function, and special notes.

Note: The function names are spelled in the uppercase and lowercase style required for C language programs.

Link-Time Libraries

At link time, any program that calls a particular API function must be linked to a library (.LIB) containing information about the function. Link-time libraries provide information that allows the operating system to dynamically link the appropriate dynamically linked library to a program at run time.

Any program calling any of the following functions must be linked to the NETAPI.LIB library:

```
NetAccessAdd
NetAccessCheck
NetAccessDel
NetAccessEnum
NetAccessGetInfo
NetAccessGetUserPerms
NetAccessSetInfo
NetAlertRaise
NetAlertStart
NetAlertStop
NetAuditClear
NetAuditRead
NetAuditWrite
NetCharDevControl
NetCharDevEnum
NetCharDevGetInfo
NetCharDevQEnum
NetCharDevQGetInfo
NetCharDevQPurge
NetCharDevQPurgeSelf
NetCharDevQSetInfo
NetConfigGet2
NetConfigGetAll2
NetConnectionEnum
NetErrorLogClear
NetErrorLogRead
NetErrorLogWrite
NetFileClose2
NetFileEnum2
NetFileGetInfo2
NetGetDCName
```

NetGroupAdd
NetGroupAddUser
NetGroupDel
NetGroupDelUser
NetGroupEnum
NetGroupGetInfo
NetGroupGetUsers
NetGroupSetInfo
NetGroupSetUsers
NetLogonEnum
NetHandleGetInfo
NetHandleSetInfo
NetMessageLogFileGet
NetMessageLogFileSet
NetMessageNameAdd
NetMessageNameDel
NetMessageNameEnum
NetMessageNameFwd
NetMessageNameGetInfo
NetMessageNameUnFwd
NetRemoteCopy
NetRemoteExec
NetRemoteMove
NetRemoteTOD
NetServerAdminCommand
NetServerDiskEnum
NetServerGetInfo
NetServerSetInfo
NetServiceControl
NetServiceEnum
NetServiceGetInfo
NetServiceInstall
NetServiceStatus
NetSessionDel
NetSessionEnum
NetSessionGetInfo
NetShareAdd
NetShareCheck
NetShareDel
NetShareEnum
NetShareGetInfo
NetShareSetInfo
NetStatisticsGet2
NetUseAdd
NetUseDel
NetUseEnum
NetUseGetInfo
NetUserAdd
NetUserDel
NetUserEnum
NetUserGetGroups
NetUserGetInfo
NetUserModalsGet
NetUserModalsSet
NetUserPasswordSet
NetUserSetGroups
NetUserSetInfo
NetUserValidate2

```
NetWkstaGetInfo
NetWkstaSetInfo
NetWkstaSetUID2
```

Any program calling any of the following functions must be linked to the NETOEM.LIB library:

```
NetMessageBufferSend
NetMessageFileSend
NetServerEnum2
```

Any program calling any of the following functions must be linked to the OS2.LIB library:

```
DosBufReset
DosCallNmPipe
DosClose
DosConnectNmPipe
DosDisconnectNmPipe
DosMakeNmPipe
DosOpen
DosPeekNmPipe
DosQNmPipeInfo
DosQNmPipeSemState
DosQNmPHandState
DosRead
DosReadAsync
DosSetNmPipeSem
DosSetNmPHandState
DosTransactNmPipe
DosWaitNmPipe
DosWrite
DosWriteAsync
SplQmAbort
SplQmClose
SplQmEndDoc
SplQmOpen
SplQmStartDoc
SplQmWrite
```

Any program calling any of the following functions must be linked to the MAILSLOT.LIB library:

```
DosDeleteMailslot
DosMailslotInfo
DosMakeMailslot
DosPeekMailslot
DosReadMailslot
DosWriteMailslot
```

Run-Time Libraries

At run time, any program that calls a particular API function must be dynamically linked to a library containing the executable binaries for that function. The OS/2 operating system automatically links the program and library together when a particular function is called.

OS/2 LAN Requester/Server provides the following dynamically linked libraries:

Library	Contents
MAILSLOT.DLL	Mailslot API library.
NETAPI.DLL	Base network API library.
NETOEM.DLL	Reserved.
SPL1A.DLL	OS/2 LAN Requester/Server extension of the Presentation Manager spooler library.

Function Notes

This section provides an alphabetical list of each OS/2 LAN Requester/Server API function, and other software required at run time.

The following list denotes special requirements for each OS/2 LAN Requester/Server API function:

W Requires *requester* service
M Requires *messenger* service
S Requires *server* service
R Can be remotely executed
A Requires administrative privileges (remote only)
L Has a local-only library available.

The following table lists each OS/2 LAN Requester/Server API function and its associated requirements.

API Name	Requirements (W M S R A L)
DosBufReset	W L
DosCallNmPipe	W L
DosClose	W L
DosConnectNmPipe	W L
DosDeleteMailslot	W L
DosDisconnectNmPipe	W L
DosDupHandle	W L
DosMailslotInfo	W L
DosMakeMailslot	W L
DosMakeNmPipe	W L
DosOpen	W L

API Name	Requirements (W M S R A L)
DosPeekMailslot	W L
DosPeekNmPipe	W L
DosQFHandState	W L
DosQHandType	W L
DosQNmPHandState	W L
DosQNmPipeInfo	W L
DosQNmPipeSemState	W L
DosRead	W L
DosReadAsync	W L
DosReadMailslot	W L
DosSetFHandState	W L
DosSetNmpHandState	W L
DosSetNmPipeSem	W L
DosTransactNmPipe	W L
DosWaitNmPipe	W L
DosWrite	W L
DosWriteAsync	W L
DosWriteMailslot	W L
NetAccessAdd	W R A
NetAccessCheck	W R
NetAccessDel	W R A
NetAccessEnum	W R A
NetAccessGetInfo	W R A
NetAccessGetUserPerms	W R A
NetAccessSetInfo	W S R A
NetAlertRaise	W
NetAlertStart	W
NetAlertStop	W
NetAuditClear	W R A
NetAuditRead	W R A
NetAuditWrite	W S
NetCharDevControl	W S R A
NetCharDevEnum	W S R
NetCharDevGetInfo	W S R
NetCharDevQEnum	W S R
NetCharDevQGetInfo	W S R

API Name	Requirements (W M S R A L)
NetCharDevQPurge	W S R A
NetCharDevQPurgeSelf	W S R
NetCharDevQSetInfo	W S R A
NetConfigGet2	W R A
NetConfigGetAll2	W R A
NetConnectionEnum	W S R A
NetErrorLogClear	W R A
NetErrorLogRead	W R A
NetErrorLogWrite	W S
NetFileClose2	W S R A
NetFileEnum2	W S R A
NetFileGetInfo2	W S R A
NetGetDCName	W R
NetGroupAdd	W R A
NetGroupAddUser	W R A
NetGroupDel	W R A
NetGroupDelUser	W R A
NetGroupEnum	W R A
NetGroupGetInfo	W R A
NetGroupGetUsers	W R A
NetGroupSetInfo	W R A
NetGroupSetUsers	W R A
NetHandleGetInfo	W R S
NetHandleSetInfo	W R S
NetLogonEnum	W R
NetMessageBufferSend	W R A
NetMessageFileSend	W R A
NetMessageLogFileGet	W M R A
NetMessageLogFileSet	W M R A
NetMessageNameAdd	W M R A
NetMessageNameDel	W M R A
NetMessageNameEnum	W M R A
NetMessageNameFwd	W M R A
NetMessageNameGetInfo	W M R A
NetMessageNameUnFwd	W M R A
NetRemoteCopy	W R

API Name	Requirements (W M S R A L)
NetRemoteExec	W R
NetRemoteMove	W R
NetRemoteTOD	W R
NetServerAdminCommand	W S R A
NetServerDiskEnum	W R A
NetServerEnum2	W R
NetServerGetInfo	W S R A
NetServerSetInfo	W S R A
NetServiceControl	W R A
NetServiceEnum	W R
NetServiceGetInfo	W R
NetServiceInstall	W R A
NetServiceStatus	W
NetSessionDel	W S R A
NetSessionEnum	W S R A
NetSessionGetInfo	W S R A
NetShareAdd	W S R A
NetShareCheck	W S R
NetShareDel	W S R A
NetShareEnum	W S R
NetShareGetInfo	W S R
NetShareSetInfo	W S R A
NetStatisticsGet2	W R A
NetUseAdd	W R A
NetUseDel	W R A
NetUseEnum	W R A
NetUseGetInfo	W R A
NetUserAdd	W R A
NetUserDel	W R A
NetUserEnum	W R A
NetUserGetGroups	W R A
NetUserGetInfo	W R A
NetUserModalsGet	W R A
NetUserModalsSet	W R A
NetUserPasswordSet	W S R A
NetUserSetGroups	W R A

API Name	Requirements (W M S R A L)
NetUserSetInfo	W R A
NetUserValidate2	W
NetWkstaGetInfo	W R A
NetWkstaSetInfo	W R A
NetWkstaSetUID2	W
SplQmAbort	R
SplQmClose	R
SplQmEndDoc	R
SplQmOpen	R
SplQmStartDoc	R
SplQmWrite	R

Appendix C. Return Codes

This appendix lists the identifiers and return codes for the OS/2 LAN Requester/Server API functions defined in the NETERR.H include file.

For a function-by-function listing of return codes, see Chapter 3, "API Function Descriptions."

Successful Return Codes

An OS/2 LAN Requester/Server API function that encounters no error returns the following code.

0 **No errors were encountered.**

Originator: NERR_SUCCESS

Redirector

An OS/2 LAN Requester/Server API function returning error codes 51 to 79 or 230 to 249 encountered an error with the redirector.

51 **This remote computer is not listening.**

Originator: ERROR_REM_NOT_LIST

52 **A duplicate name exists on the network.**

Originator: ERROR_DUP_NAME

53 **The network path cannot be found.**

Originator: ERROR_BAD_NETPATH

54 **The network is busy.**

Originator: ERROR_NETWORK_BUSY

55 **This device does not exist on the network.**

Originator: ERROR_DEV_NOT_EXIST

56 **The IBM NETBIOS command limit has been exceeded.**

Originator: ERROR_TOO_MANY_CMDS

57 **A network adapter hardware error has occurred.**

Originator: ERROR_ADAP_HDW_ERR

58 **The network has responded incorrectly.**

Originator: ERROR_BAD_NET_RESP

59 **An unexpected network error has occurred.**

Originator: ERROR_UNEXP_NET_ERR

60 **The remote adapter being used is incompatible.**

Originator: ERROR_BAD_REM_ADAP

61 **The print queue is full.**

Originator: ERROR_PRINTQ_FULL

62 **There is not enough memory available for the requested print file.**

Originator: ERROR_NO_SPOOL_SPACE

63 **The requested print file has been canceled.**

Originator: ERROR_PRINT_CANCELLED

64 **The network name was deleted.**

Originator: ERROR_NETNAME_DELETED

65 **Network access is denied.**

Originator: ERROR_NETWORK_ACCESS_DENIED

66 **This network device type is incorrect.**

Originator: ERROR_BAD_DEV_TYPE

67 **This network name cannot be found.**

Originator: ERROR_BAD_NET_NAME

68 **The network name limit has been exceeded.**

Originator: ERROR_TOO_MANY_NAMES

69 **The IBM NETBIOS session limit has been exceeded.**

Originator: ERROR_TOO_MANY_SESS

70 **File sharing has been temporarily paused.**

Originator: ERROR_SHARING_PAUSED

71 **This request is not accepted by the network.**

Originator: ERROR_REQ_NOT_ACCEP

72 **Print or disk redirection is temporarily paused.**

Originator: ERROR_REDIR_PAUSED

230 **This is a non-existent pipe or an invalid operation**

Originator: ERROR_BAD_PIPE

231 **The specified pipe is busy.**

Originator: ERROR_PIPE_BUSY

232 **There is no data on a non-blocking read.**

Originator: ERROR_NO_DATA

233 **The pipe was disconnected by the server.**

Originator: ERROR_PIPE_NOT_CONNECTED

234 **Additional data is available, but the buffer is too small.**

Originator: ERROR_MORE_DATA

240 **The session was canceled.**

Originator: ERROR_VC_DISCONNECTED

Network Utilities

An OS/2 LAN Requester/Server API function returning error codes 2100 to 2140 encountered an error with a network utility.

2102 **The redirector NETWKSTA.EXE has not been started.**

Originator: NERR_NetNotStarted

2103 **The server cannot be located.**

Originator: NERR_UnknownServer

2104 **An internal error occurred—the network cannot access a shared memory segment.**

Originator: NERR_ShareMem

2105 **A network resource shortage occurred.**

Originator: NERR_NoNetworkResource

2106 **This operation is not supported on workstations.**

Originator: NERR_RemoteOnly

2107 **The device is not connected.**

Originator: NERR_DevNotRedirected

2108 **Unknown local device.**

Originator: NERR_UnknownLocalDev

2114 **The Server service has not been started.**

Originator: NERR_ServerNotStarted

2115 **The device queue is empty.**

Originator: NERR_ItemNotFound

2116 **The device or directory does not exist.**

Originator: NERR_UnknownDevDir

2117 **The operation is invalid on a redirected device.**

Originator: NERR_RedirectedPath

2118 **The name has already been shared.**

Originator: NERR_DuplicateShare

2119 **The server is currently out of the requested resource.**

Originator: NERR_NoRoom

2121 The requested add of item exceeds maximum allowed.

Originator: NERR_TooManyItems

2123 The buffer is too small for fixed-length data.

Originator: NERR_BufTooSmall

2127 A remote API error has occurred.

Originator: NERR_RemoteErr

2131 An error occurred when opening or reading the IBMLAN.INI file.

Originator: NERR_LanIniError

2134 An internal error occurred when calling the workstation driver.

Originator: NERR_OS2IoctlError

2136 A general network error has occurred.

Originator: NERR_NetworkError

2138 The Requester service has not been started.

Originator: NERR_WkstaNotStarted

2139 The requested information is not available.

Originator: NERR_BrowserNotStarted

2140 An internal error has occurred.

Originator: NERR_InternalError

Spooler

An OS/2 LAN Requester/Server API function returning error codes 2150 to 2179 encountered an error while modifying a network spooler.

2150 The printer queue does not exist.

Originator: NERR_QNotFound

2151 The print job does not exist.

Originator: NERR_JobNotFound

2152 The printer destination cannot be found.

Originator: NERR_DestNotFound

2154 The printer queue already exists.

Originator: NERR_QExists

2155 No more printer queues can be added.

Originator: NERR_QNoRoom

2156 No more print jobs can be added.

Originator: NERR_JobNoRoom

2157 No more printer destinations can be added.

Originator: NERR_DestNoRoom

2158 This printer destination is idle and cannot accept control operations.

Originator: NERR_DestIdle

2159 This printer destination request contains an invalid control function.

Originator: NERR_DestInvalidOp

2160 The printer processor is not responding.

Originator: NERR_ProcNoRespond

2161 The spooler service has not been started.

Originator: NERR_SpoolerNotLoaded

2163 This operation cannot be performed on the printer queue in its current state.

Originator: NERR_QInvalidState

2164 This operation cannot be performed on the print job in its current state.

Originator: NERR_JobInvalidState

2165 A spooler memory allocation failure has occurred.

Originator: NERR_SpoolNoMemory

Service

An OS/2 LAN Requester/Server API function returning error codes 2180 to 2199 encountered an error with one of the network services.

2180 The service does not respond to control actions.

Originator: NERR_ServiceTableLocked

2181 The service table is full.

Originator: NERR_ServiceTableFull

2182 The requested service has already been started.

Originator: NERR_ServiceInstalled

2183 The service does not respond to control actions.

Originator: NERR_ServiceEntryLocked

2184 The service has not been started.

Originator: NERR_ServiceNotInstalled

2185 The service name is invalid.

Originator: NERR_BadServiceName

2186 The service is not responding to the control function.

Originator: NERR_ServiceCtlTimeout

2187 The service control is busy.

Originator: NERR_ServiceCtlBusy

2188 The IBMLAN.INI file contains an invalid service program name.

Originator: NERR_BadServiceProgName

2189 The service cannot be controlled in its present state.

Originator: NERR_ServiceNotCtrl

2190 The service was ended abnormally.

Originator: NERR_ServiceKillProc

2191 The requested pause or stop is not valid for this service.

Originator: NERR_ServiceCtlNotValid

Requester

An OS/2 LAN Requester/Server API function returning error codes 2200 to 2219 encountered an error with a requester.

2200 This workstation is already logged on.

Originator: NERR_AlreadyLoggedOn

2201 This workstation has not been logged on yet.

Originator: NERR_NotLoggedOn

2202 The user name or group name parameter is invalid.

Originator: NERR_BadUsername

2203 The password parameter is invalid.

Originator: NERR_BadPassword

2204 The logon processor did not add the message alias.

Originator: NERR_UnableToAddName_W

2205 The logon processor did not add the message alias.

Originator: NERR_UnableToAddName_F

2206 The logoff processor did not delete the message alias.

Originator: NERR_UnableToDelName_W

2207 The logoff processor did not delete the message alias.

Originator: NERR_UnableToDelName_F

2210 A centralized logon server conflict has occurred.

Originator: NERR_LogonServerConflict

2211 The server is configured without a valid user path.

Originator: NERR_LogonNoUserPath

2212 An error occurred while loading or running the logon script.

Originator: NERR_LogonScriptError

2213 Unable to use resources.

Originator: NERR_CentralLogonFailed

2214 The logon server was not specified—standalone logon will occur.

Originator: NERR_StandaloneLogon

2215 The logon server cannot be found.

Originator: NERR_LogonServerNotFound

2216 There is already a logon domain for this computer.

Originator: NERR_LogonDomainExists

2217 The logon server could not validate the logon.

Originator: NERR_NonValidatedLogon

Access, User, and Group

An OS/2 LAN Requester/Server API function returning error codes 2220 to 2249 encountered an error while requesting or modifying information concerning network privileges.

2220 The group does not exist.

Originator: NERR_GroupNotFound

2221 The user name cannot be found.

Originator: NERR_UserNotFound

2222 The netname cannot be found.

Originator: NERR_ResourceNotFound

2223 The group name is already in use.

Originator: NERR_GroupExists

2224 The user account already exists.

Originator: NERR_UserExists

2225 **The resource permission list already exists.**

Originator: NERR_ResourceExists

2226 **The UAS database is replicant and will not allow updates.**

Originator: NERR_NotPrimary

2227 **The UAS database has not been started.**

Originator: NERR_ACFNotLoaded

2228 **There are too many names in the access control file.**

Originator: NERR_ACFNoRoom

2229 **An error was encountered in accessing the accounts database.**

Originator: NERR_ACFFileIOFail

2230 **Too many lists were specified.**

Originator: NERR_ACFTooManyLists

2231 **Deleting a user with a session is not allowed.**

Originator: NERR_UserLogon

2232 **The parent directory cannot be located.**

Originator: NERR_ACFNoParent

2233 **Unable to grow UAS sess cache segment.**

Originator: NERR_CanNotGrowSegment

2234 **This operation is not allowed on this special group.**

Originator: NERR_SpeGroupOp

2235 **This user is not cached in UAS sess cache.**

Originator: NERR_NotInCache

2236 **The user already belongs to this group.**

Originator: NERR_UserInGroup

2237 The user does not belong to this group.

Originator: NERR_UserNotInGroup

2238 The user account is undefined.

Originator: NERR_AccountUndefined

2239 The user account has expired.

Originator: NERR_AccountExpired

2240 The user is not allowed to log on from this requester.

Originator: NERR_InvalidWorkstation

2241 The user is not allowed to log on at this time.

Originator: NERR_InvalidLogonHours

2242 The password has expired.

Originator: NERR_PasswordExpired

2243 This password cannot change.

Originator: NERR_PasswordCantChange

2244 This password cannot be used now.

Originator: NERR_PasswordHistConflict

2245 The password is shorter than required.

Originator: NERR_PasswordTooShort

2246 The password is too recent to change.

Originator: NERR_PasswordTooRecent

2247 The UAS database file is corrupted.

Originator: NERR_InvalidDatabase

2248 No updates are neccessary to this replicant.

Originator: NERR_Data2249UpToDate

Use

An OS/2 LAN Requester/Server API function returning error codes 2250 to 2269 encountered an error while trying to retrieve information about a resource or while using a resource.

2250 The connection cannot be found.

Originator: NERR_UseNotFound

2251 This asg_type is invalid.

Originator: NERR_BadAsgType

2252 This device is already being shared.

Originator: NERR_DeviceIsShared

Message

An OS/2 LAN Requester/Server API function returning error codes 2270 to 2309 encountered an error while processing information about a server.

2270 A computer name has not been configured.

Originator: NERR_NoComputerName

2271 This message server has already been started.

Originator: NERR_MsgAlreadyStarted

2272 The message server initialization request has failed.

Originator: NERR_MsgInitFailed

2273 The message alias cannot be found on the local area network.

Originator: NERR_NameNotFound

2274 This message alias has already been forwarded.

Originator: NERR_AlreadyForwarded

2275 This message alias has been added but is still forwarded.

Originator: NERR_AddForwarded

2276 This message alias already exists locally.

Originator: NERR_AlreadyExists

2277 The maximum number of added message aliases has been exceeded.

Originator: NERR_TooManyNames

2278 The computer name cannot be deleted.

Originator: NERR_DelComputerName

2279 Messages cannot be forwarded back to the same workstation.

Originator: NERR_LocalForward

2280 Error in domain message processor.

Originator: NERR_GrpMsgProcessor

2281 The message has been sent but the reception is currently paused.

Originator: NERR_PausedRemote

2282 The message was sent but not received.

Originator: NERR_BadReceive

2283 The message alias is currently in use—try again later.

Originator: NERR_NameInUse

2284 The messenger service has not been started.

Originator: NERR_MsgNotStarted

2285 The name is not on the local computer.

Originator: NERR_NotLocalName

2286 The forwarded message alias cannot be found on the network.

Originator: NERR_NoForwardName

2287 The message alias table on the remote station is full.

Originator: NERR_RemoteFull

2288 Messages for this alias are not currently forwarded.

Originator: NERR_NameNotForwarded

2289 The broadcast message was truncated.

Originator: NERR_TruncatedBroadcast

2290 An error occurred in reading the message file.

Originator: NERR_FileError

2294 This is an invalid device.

Originator: NERR_InvalidDevice

2295 A write fault has occurred.

Originator: NERR_WriteFault

2297 A duplicate message alias exists on the local area network.

Originator: NERR_DuplicateName

2298 This message alias will be deleted later.

Originator: NERR_DeleteLater

2299 The message alias was not successfully deleted from all networks.

Originator: NERR_IncompleteDel

2300 This operation is not supported on machines with multiple networks.

Originator: NERR_MultipleNets

Server

An OS/2 LAN Requester/Server API function returning error codes 2310 to 2329 encountered an error when processing information about a server.

2310 This shared resource does not exist.

Originator: NERR_NetNameNotFound

2311 This device is not shared.

Originator: NERR_DeviceNotShared

2312 **A session does not exist with that computer name.**

Originator: NERR_ClientNameNotFound

2314 **There is not an open file with that ID number.**

Originator: NERR_FileIdNotFound

2315 **A failure occurred when executing a remote administration command.**

Originator: NERR_ExecFailure

2316 **A failure occurred when opening a remote temporary file.**

Originator: NERR_TmpFile

2317 **The data returned from a remote administration command has been truncated to 64KB.**

Originator: NERR_TooMuchData

2318 **This device cannot be shared as both a spooled and a non-spooled device.**

Originator: NERR_DeviceShareConflict

2319 **The server table was initialized incorrectly.**

Originator: NERR_BrowserTableIncomplete

2320 **This domain is not active on this computer.**

Originator: NERR_NotLocalDomain

Serial Device

An OS/2 LAN Requester/Server API function returning error codes 2330 to 2349 encountered an error with a serial device.

2331 **The operation is invalid for this device.**

Originator: NERR_DevInvalidOpCode

2332 **This device cannot be shared.**

Originator: NERR_DevNotFound

2333 **This device was not open.**

Originator: NERR_DevNotOpen

2334 This device name string is invalid.

Originator: NERR_BadQueueDevString

2335 The queue priority is invalid.

Originator: NERR_BadQueuePriority

2337 There are no shared communication devices.

Originator: NERR_NoCommDevs

2338 A queue does not exist for this request.

Originator: NERR_QueueNotFound

2340 This list of devices is invalid.

Originator: NERR_BadDevString

2341 The requested device is invalid.

Originator: NERR_BadDev

2342 This device is already in use by the spooler.

Originator: NERR_InUseBySpooler

2343 This device is already in use as a communications device.

Originator: NERR_CommDevInUse

I/O

An OS/2 LAN Requester/Server API function returning error codes 2350 to 2369 encountered an error while processing input or output.

2351 The specified computer name is invalid.

Originator: NERR_InvalidComputer

2354 The string and prefix specified are too long.

Originator: NERR_MaxLenExceeded

2356 This path component is invalid.

Originator: NERR_BadComponent

2357 The type of input cannot be determined.

Originator: NERR_CantType

2362 The buffer for types is not big enough.

Originator: NERR_TooManyEntries

Audit Log and Error Log

An OS/2 LAN Requester/Server API function returning error codes 2377 to 2379 encountered an error writing or reading from the audit log file or error log file.

2377 This log file exceeds the maximum defined size.

Originator: NERR_LogOverflow

2378 This log file has changed between reads.

Originator: NERR_LogFileChanged

2379 This log file is corrupt.

Originator: NERR_LogFileCorrupt

Remote Error

An OS/2 LAN Requester/Server API function returning error codes 2380 to 2399 encountered an error while executing a remote process.

2380 The source path cannot be a directory.

Originator: NERR_SourceIsDir

2381 The source path is illegal.

Originator: NERR_BadSource

2382 The destination path is illegal.

Originator: NERR_BadDest

2383 The source and destination paths are on different servers.

Originator: NERR_DifferentServers

2385 The run server you requested using the NET RUN command is paused.

Originator: NERR_RunSrvPaused

2386 An error was detected while creating a thread.

Originator: NERR_CreatingThread

2387 An error was detected while creating a pipe.

Originator: NERR_ErrorMakingPipe

2389 An error occurred when communicating with a run server.

Originator: NERR_ErrCommRunSrv

2390 An error occurred when connecting to run server.

Originator: NERR_ErrConnRunSrv

2391 An error occurred when starting a background process.

Originator: NERR_ErrorExecingGhost

2392 The shared resource you are connected to could not be found.

Originator: NERR_ShareNotFound

Requester Redirector

An OS/2 LAN Requester/Server API function returning error codes 2400 to 2429 encountered an error with the requester redirector.

2400 The LAN adapter number is invalid.

Originator: NERR_InvalidLana

2401 There are open files on the connection.

Originator: NERR_OpenFiles

2402 Active connections still exist.

Originator: NERR_ActiveConns

2403 This netname or password is invalid.

Originator: NERR_BadPasswordCore

2404 The device is being accessed by an active process.

Originator: NERR_DevInUse

2405 **The drive letter is in use locally.**

Originator: NERR_LocalDrive

2406 **Cannot allocate sufficient memory to load requester software.**

Originator: NERR_MemAllocMsg

2407 **Error reading NETWORKS entry in the IBMLAN.INI file.**

Originator: NERR_IniFilRdErr

2408 **Too many NETWORKS entries in the IBMLAN.INI file.**

Originator: NERR_MultNetsMsg

2409 **NETWORKS entry in the IBMLAN.INI file is too long and is ignored**

Originator: NERR_BadNetEntHdr

2410 **Error opening a network device driver.**

Originator: NERR_BadBiosMsg

2411 **There has been an improper BiosLinkage response from a device driver.**

Originator: NERR_BadLinkMsg

2412 **Argument given to the function is not valid.**

Originator: NERR_BadArgMsg

2413 **Incorrect OS/2 version in use.**

Originator: NERR_BadVerMsg

2414 **A redirector is already started.**

Originator: NERR_RdrInstMsg

2415 **Drive in local use.**

Originator: NERR_LocalDrive

2416 **Error starting the NETWKSTA.SYS.**

Originator: NERR_Version

2430 The specified client is already registered for the specified event.

Originator: NERR_AlertExists

2431 The Alerter service table is full.

Originator: NERR_TooManyAlerts

2432 The Alerter service has not been started.

Originator: NERR_NoSuchAlert

2433 The Alerter service recipient is invalid.

Originator: NERR_BadRecipient

2440 The log file does not contain the requested record number.

Originator: NERR_InvalidLogSeek

2450 The UAS data2450 is not configured correctly.

Originator: NERR_BadUasConfig

2455 The Netlogon service has not been started.

Originator: NERR_NetLogonNotStarted

2456 It is not possible to grow the UAS file.

Originator: NERR_CanNotGrowUASFile

Appendix D. Creating OS/2 LAN Server Services

When designing a service to use with the OS/2 LAN Requester/Server software, keep in mind the following requirements:

- An executable file of a service must be listed under the SERVICE component section in the IBMLAN.INI file.
- A service must not call screen or keyboard functions; this can be done indirectly by calling a pop-up function such as VioPopup (an OS/2 program function).
- A service must dynamically notify the OS/2 LAN Requester/Server software about a change in its status by calling the NetServiceStatus function, so that other applications can respond correctly to the change.
- A service must respond to any signal sent by an application, calling the NetServiceControl function to change its current state of operation.

An IBMLAN.INI entry of a service must include the name of the service and a valid path name of an executable file used to start the service, and can have additional parameters supply other information. Applications requesting to use the service can call the functions in the Configuration category to obtain the additional information. For information on using the Configuration functions and a description of IBMLAN.INI components, see "Configuration Category" on page 3-68.

Starting a Service

After a service is started on a computer, the service must:

1. Call the NetServiceInstall function to notify the OS/2 LAN Requester/Server software that a service is being started.
2. Verify any command-line parameters passed to the service by the calling process.
3. Start a signal-handler to interpret opcodes that requesting applications pass.
4. Set its state to INSTALL_PENDING to notify any requesting applications that it is not ready for use, by calling the NetServiceStatus function.
5. Complete any other initialization procedures previously defined by the service.
6. Notify the OS/2 LAN Requester/Server software that installation is complete by calling the NetServiceStatus function and setting its state to INSTALLED.

NetServiceInstall executes the executable file specified in the IBMLAN.INI component of the service by calling the OS/2 DosExecPgm function and passing the string of parameters comprised of IBMLAN.INI parameters and information passed to the *cmdargs* parameter of NetServiceInstall.

The DosExecPgm function executes the service in detached mode, preventing handles from being passed to child processes and preventing screen- and keyboard-oriented calls except through pop-up functions. A service inherits the environment of the parent process—the NetServiceInstall function.

If a service includes more than one process, the process that DosExecPgm initially executes, referred to as the main service process, is the only process that receives standard signals from NetServiceControl. The main service process is the only one that can issue calls to the NetServiceStatus function. A service can transfer the

responsibilities of the main service process to another process by setting the pid component of the *service_status* data structure (passed to NetServiceStatus) to the process identification number (PID) of the main service process candidate.

After receiving control from the DosExecPgm function, a service validates the parameters passed from the IBMLAN.INI file. If the parameters are invalid, a service notifies the OS/2 LAN Requester/Server software by calling the NetServiceStatus function and ending execution. Otherwise, the service installation continues.

After verifying parameters, a service must start a signal-handler that communicates with the NetServiceControl function. Using the signal-handler, a service specifies its current state (such as INSTALL_PENDING or INSTALLED) for the OS/2 LAN Requester/Server software to enable other applications to properly use the service. The signal-handler must register a function for the FlagA signal, opcode SERVICE_REC_SIG_FLAG by calling the DosSetSigHandler function.

The following pseudocode illustrates the general format for a signal-handler:

```
void far pascal
signal_handler (sig_arg, sig_no)
unsigned sig_arg;
unsigned sig_no;
{
struct service_status svci;

unsigned char opcode; /* opcode parameter from
                         the NetServiceControl function */
unsigned char arg;    /* arg parameter from
                         the NetServiceControl function */

opcode = (unsigned char) (sig_arg & 0xff);
arg    = (unsigned char) ((sig_arg) >> 8) & 0xff);

/* set up default values for NetServiceStatus buffer */

svci.svcs_pid = 0;
svci.svcs_status = SERVICE_INSTALLED | SERVICE_UNINSTALLABLE;
svci.svcs_code = 0L;

switch (opcode) {

   case SERVICE_CTRL_INTERROGATE :

   set_wksta_status (& svci);
   break;

/* other opcodes as appropriate
   .
   .
   .
*/
```

```
case SERVICE_CTRL_UNINSTALL :

   svci.svcs_status = SERVICE_UNINSTALL_PENDING;
   set_wksta_status = (& svci);

   Terminate ();

default :

   /* treat all unknown commands
     as SERVICE_CTRL_INTERROGATE */
}

/* issue a 'reset' for the signal */

DosSetSigHandler (0, 0, 0, SIG_RESET, sig_no);

return;


/*
   Note that if the signal-handler is written in C, it must be
   compiled using the auto-load DS option (-A option).
   Otherwise, the DS register will not be loaded to the
   service's default data segment when the signal-handler
   is called, and the code in the handler may make incorrect
   assumptions about the data's location.

   Signal-handlers must preserve registers as noted
   in the OS/2 programming documentation.

*/
}
```

After a service specifies its current state as INSTALLED, another application can change or query the state of the service by calling NetServiceControl. With this function, an application can pass an opcode specifying one of four actions to take, as defined in SERVICE.H, as follows:

Manifest	Value	Meaning
SERVICE_CTRL_INTERROGATE	0	Request for general information.
SERVICE_CTRL_PAUSE	1	Pause the service.
SERVICE_CTRL_CONTINUE	2	Continue a paused service.
SERVICE_CTRL_UNINSTALL	3	Shut a service down.

A service can define its own set of valid opcodes. Invalid opcodes should default to another opcode such as SERVICE_CTRL_INTERROGATE. The OS/2 LAN Requester/Server software defines the following limits:

- A service that does not accept the SERVICE_UNINSTALL opcode cannot be removed at any time.

- While in the SERVICE_INSTALL_PENDING state, a service can receive only the SERVICE_UNINSTALL opcode.

In the following pseudocode, an application and a service communicate by means of a signal-handler and the OS/2 LAN Requester/Server software:

```
Application OS/2 LAN Requester/Server Service
----------- ----------- -------

application calls
NetServiceControl
specifying an
opcode to perform
a particular task
|
|  opcode is sent
+--->>------ to the specified
service
  |
  |  signal-handler
  +---->>----- interprets opcode
     |
   performs the task
   defined by the opcode
    |
   calls NetServiceStatus
   to change the state of
   the service, if required
    |
 updates service ---<<---+
 information table
 according to current
 state of the service
  |
  |
NetServiceControl |
returns with the -<<-+
appropriate information
about the service's
current state
```

If performing the task takes a long time (more than a few seconds), the service should make intermediate calls to the NetServiceStatus function.

Stopping a Service

When a service is no longer needed, either the application using it or the service itself should call NetServiceControl and specify the appropriate shut-down opcode. After receiving a shut-down opcode, a service must again call NetServiceStatus to declare SERVICE_UNINSTALL_PENDING status and then perform any other necessary tasks, such as closing open resources. The final step is to call the OS/2 DosExit function. Immediately before doing this, the service must again change its status, this time to SERVICE_UNINSTALLED.

For an application to disable a service from processing any further requests, the service performs the following steps:

- Call NetServiceStatus to set a SERVICE_UNINSTALL_PENDING state
- Execute a cleanup routine, closing any open resources
- Notify the OS/2 LAN Requester/Server software that it has been removed by calling NetServiceStatus and setting a SERVICE_UNINSTALLED state
- End program execution.

To obtain information about services started, an application calls NetServiceEnum. To query the state of a service, retrieving its status and code information, an application calls NetServiceStatus.

The following pseudocode illustrates how the alerter service communicates with other services, applications, and the OS/2 LAN Requester/Server software:

```
spooler (print alerts) -->>---+
|
server (security alerts) ->>--+
|
   NetAlertRaise -->>-+
| |
Other Services ------>>-------------+ |
| |
Other Applications ---->>------------+ |
|
     local mailslot
|
text event |
NetMessageBufferSend -<<- ALERTER Service ---<<----+
|
|
+--------->>-----------+
|
|
 Remote Messenger
   Service
```

For information on the default OS/2 LAN Requester/Server services and the functions used to control services, see "Service Category" on page 3-298.

Appendix E. OS/2 LAN API Support under IBM DOS Requesters

Chapter 3, "API Function Descriptions," contains detailed descriptions of the OS/2 LAN Requester/Server API functions. This appendix presents information programmers should be aware of when using OS/2 LAN Requester/Server APIs with DOS Requester.

In this appendix, you will find information about:

- IBM DOS API services
- IBM DOS libraries, where they are stored, and which ones are needed
- API functions supported under IBM DOS and any differences in their use.

Note: When porting OS/2 LAN Requester/Server applications to run under DOS, be aware that DOS, unlike the OS/2 program, does not support pointer checking, semaphores, or shared memory segments. Also note that all file names, directory names, or parts of a path name, including UNC server and share names, must follow DOS naming conventions.

API Services Supported Under DOS

DOS Requester supports the following services:

Service	Purpose
messenger	Enables applications to send messages across the LAN.
netpopup	Enables messages to be displayed as pop-up messages on the screen.
requester	Enables DOS computers (attached to the LAN) to be configured as requesters, thus enabling them to access resources on remote OS/2 LAN servers.

These services are installed during installation of the redirector program. Unlike the OS/2 program, the services cannot be completely stopped once they are installed. However, they can be paused and continued using the NetServiceControl function.

DOS API Libraries

The OS/2 program uses dynamically-linked libraries during run time; DOS does not. Under DOS, link your application with either the DOSNET.LIB or WINNET.LIB library and the SYSCALL0.LIB library.

The DOSNET.LIB and SYSCALL0.LIB libraries also serve as BIND libraries for the OS/2 NETAPI.LIB, NAMEPIPE.LIB, MAILSLOT.LIB, and NETSPOOL.LIB libraries (similar to the API.LIB library).

The following sample commands show which DOS libraries to link in with your application to achieve the same functionality that NETAPI.LIB provides:

```
LINK mydosap.obj, mydosap.exe, mydosap.map /MAP,
c:\ibmlan.dos\netsrc\lib\dosnet.lib

BIND myos2ap.exe doscalls.lib c:\ibmlan.dos\netsrc\lib\netapi.lib
c:\ibmlan.dos\netsrc\lib\dosnet.lib
c:\ibmlan.dos\netsrc\lib\syscall0.lib api.lib
```

The SYSVALL0.LIB library file supports networking functions when using the OS/2 FAPI system calls (DosOpen, DosBufReset, DosRead). If used, it must immediately precede the API.LIB file name as shown in the preceding BIND command line.

The default storage location for *dosnet.lib, syscall0.lib*, and *winnet.lib* is

```
c:\ibmlan.dos\netsrc\lib.
```

Include Files

DOS Requester uses most of the OS/2 include files described in Appendix A, "Include Files." The include files are stored by default in

```
c:\ibmlan.dos\netsrc\h.
```

The following include files are used under DOS:

API Category	Include Files	API Category	Include Files
Access Permission	NETCONS.H ACCESS.H	Profile	NETCONS.H PROFILE.H
Configuration	NETCONS.H CONFIG.H	Remote Utility	NETCONS.H REMUTIL.H
Mailslot	NETCONS.H MAILSLOT.H	Server	NETCONS.H SERVER.H
Message	NETCONS.H MESSAGE.H	Service	NETCONS.H SERVICE.H
Session	NETCONS.H SHARES.H		
Share	NETCONS.H SHARES.H ACCESS.H	Named Pipe	NETCONS.H NMPIPE.H
Use	NETCONS.H USE.H	User	NETCONS.H ACCESS.H
Requester	NETCONS.H WKSTA.H ACCESS.H		

Differences in Use Under DOS

DOS Requester supports a subset of the API functions described in Chapter 3, "API Function Descriptions." Most of the DOS Requester API functions are executed on a remote server. API functions that can be executed remotely must include a remote server name parameter to identify where the function is to be executed. Attempting to execute a remote only function on a local requester returns NERR_RemoteOnly.

Those functions that can be executed on a local requester must be called with a NULL server name parameter (defaults to local requester name) or the name assigned the local requester in the format–computer name. Attempting to execute a local only function on a remote server returns ERROR_NOT_SUPPORTED.

The following sections describe the API functions that DOS LAN Requester supports. In the descriptions, both the categories and the functions within each category are alphabetically listed along with any differences in their use from that described in Chapter 3, "API Function Descriptions."

Access Permission

The functions in the Access Permission category examine or modify user or group access permission records for server resources. Under DOS, these functions can only be executed on a remote server. Administrative privilege must have been granted to execute the functions. Attempting to execute the functions on a local requester returns NERR_RemoteOnly.

Function	Differences in Use
NetAccessAdd (*admin*)	None
NetAccessDel (*admin*)	None
NetAccessEnum (*partially admin*)	None
NetAccessGetInfo (*partially admin*)	None
NetAccessGetUserPerms (*partially admin*)	
NetAccessSetInfo (*admin*)	None

Auditing

Function	Differences in Use
NetAuditClear (local)	None

Configuration

The functions in the Configuration category cannot retrieve network configuration information from the DOSLAN.INI file. However, these functions can be called remotely on a DOS requester to retrieve information from the IBMLAN.INI file.

Function	Differences in Use
NetConfigGet2 (admin, DOS)	Cannot be called locally on a DOS requester
NetConfigGetAll2 (admin, DOS)	Cannot be called locally on a DOS requester

Connection

Function	Differences in Use
NetConnectionEnum (local)	None

Error Logging

Function	Differences in Use
NetErrorLogClear (local)	None
NetErrorLogRead (local)	

File

Function	Differences in Use
NetFileClose2 (local)	None
NetFileEnum2 (local)	None
NetFileGetInfo2	

Group

Function	Differences in Use
NetGroupAdd (local)	None
NetGroupAddUser (local)	None
NetGroupDel (local)	None
NetGroupDelUser (local)	None
NetGroupEnum (local)	None
NetGroupGetUsers (local)	None

Mailslot

The functions in the Mailslot category provide one-way interprocess communication (IPC). Under DOS, the functions can be executed on a local requester or remote server.

Note that mailslots can only be read or deleted by the process that created them. Mailslots created by a process are deleted when that process ends.

Function	Differences in Use
DosDeleteMailslot	None
DosMailslotInfo	None
DosMakeMailslot	None
DosPeekMailslot	None
DosReadMailslot	None
DosWriteMailslot	None

Message

The functions in the Message category are used to send and receive messages. The functions can be executed only on a local requester. Attempting to execute the functions on a remote server returns ERROR_NOT_SUPPORTED.

Under DOS, messages cannot be forwarded, unforwarded, or logged.

By default, DOS LAN Requester accepts only two names in the message name table: the name of the requester and the name of the user. To define more names, edit the DOSLAN.INI file and change the value of the *nmsg* parameter for the messenger component. For more information on the DOSLAN.INI file, see the *DOS User's Guide*.

The maximum size of a message under DOS is 64KB.

Function	Differences in Use
NetMessageBufferSend (local)	Under DOS, the name parameter cannot point to the name of the local requester or to the users currently logged on to that requester.
NetMessageFileSend (local)	Under DOS, the name parameter cannot point to the name of the local requester or to the users currently logged on to that requester.
NetMessageLogFileGet (local)	None
NetMessageLogFileSet (local)	None
NetMessageNameAdd (local)	None
NetMessageNameDel (local)	None
NetMessageNameEnum (local)	None
NetMessageNameFwd (local)	None

Function	Differences in Use
NetMessageNameGetInfo (local)	None
NetMessageNameUnFwd (local)	None

Named Pipe

The functions in the Named Pipe category control interprocess communication (IPC) for named pipes. The functions can be executed only on a remote server which has interprocess communication shares.

DOS supports only client processes; a pipe must have already been created and connected on a remote server. Child processes inherit the open file handles of the parent processes. DOS does not support asynchronous reading and writing of named pipes.

Note: The FAPI replacement library routine for DosOpen provides support for DASD opens (open Mode Flag 0x8000). Since DOS does not support this operation, pipe operations on this type of file handle return ERROR_INVALID_HANDLE rather than ERROR_BAD_PIPE.

Function	Differences in Use
DosBufReset	DosBufReset works differently depending on the version of DOS you are programming under. Under versions 3.3 and 4.0, DosBufreset returns 0 after resetting a closed named pipe. If the handle is to a named pipe that has already been closed, DosBufReset returns ERROR_BROKEN_PIPE. Under DOS 3.3 and 4.0, DosBufReset waits for the pipe to be emptied before resetting it.
DosCallNmPipe	None
DosClose	None
DosDupHandle	None
DosOpen	None
DosPeekNmPipe	None
DosQHandType	None
DosQNmpHandState	None
DosQNmPipeInfo	None
DosRead	None
DosSetNmpHandState	None
DosTransactNmPipe	None
DosWrite	None

Remote Utility

The functions in the Remote Utility category enable applications to copy and move remote files, and access the time-of-day information on a remote server. Attempting to execute the functions on a local requester returns NERR_RemoteOnly.

Function	Differences in Use
NetRemoteCopy (Server)	None
NetRemoteMove (Server)	None
NetRemoteTOD (Server)	None

Requester

The functions in the Requester category control the operation of requesters. They can be executed only on a local requester. Attempting to execute the functions on a remote server returns ERROR_NOT_SUPPORTED.

Function	Differences in Use
NetWkstaGetInfo (local)	None
NetWkstaSetInfo (local)	None
NetWkstaSetUID2 (local)	None

Certain parameters are not used under DOS and therefore cannot be set. However, validity checks are performed on most of the unused parameters for future expansion. The following table describes the parameters that are used and indicates whether validity checks are performed on them:

Parameter	Used	Validity Checked
charwait	No	Yes
chartime	Yes	Yes
charcount	Yes	Yes
errlogsz	No	Yes
printbuftime	No	Yes
wrkheuristics	No	No

Serial Device

Function	Differences in Use
NetCharDevControl (local)	None
NetCharDevEnum (local)	None
NetCharDevQEnum (local)	None
NetCharDevQGetInfo (local)	None
NetCharDevQPurge (local)	None

Function	Differences in Use
NetCharDevQPurgeSelf (local)	None
NetCharDevQSetInfo (local)	None

Server

The functions in the Server category enable remote administration tasks to be performed on a remote server. NetServerEnum can be executed on either a local requester or remote server; all other server functions are executed on a remote server. Attempting to execute NetServerAdminCommand or NetServerGetInfo on a local requester returns NERR_RemoteOnly.

Function	Differences in Use
NetServerAdminCommand (Server)	None
NetServerDiskEnum (Server, local)	Usually called locally (same as the OS/2 program)
NetServerEnum2	None
NetServerGetInfo (Server)	None
NetServerSetInfo (Server)	None

Service

The functions in the Service category control network service programs. They are executed on a local requester. Attempting to execute the functions on a remote server returns ERROR_NOT_SUPPORTED.

Under DOS, the services cannot be completely stopped; however, they can be paused and continued using NetServiceControl.

Function	Differences in Use
NetServiceControl (local)	None
NetServiceEnum (local)	None
NetServiceGetInfo (local)	None
NetServiceInstall (local)	None

Session

Function	Differences in Use
NetSessionDel	None
NetSessionEnum	None
NetSessionGetInfo	None

Share

The functions in the Share category control shared resources. They can be executed only on a remote server. Attempting to execute the functions on a local requester returns NERR_RemoteOnly.

Function	Differences in Use
NetShareAdd	None
NetShareCheck	None
NetShareDel	None
NetShareEnum	None
NetShareGetInfo	None
NetShareSetInfo	None

Statistics

Function	Differences in Use
NetStatisticsGet2	None

Use

The functions in the Use category examine or control connections (uses) between requesters and servers. They can be executed only on a remote server. Attempting to execute the functions on a local requester returns NERR_RemoteOnly.

Function	Differences in Use
NetUseAdd (local)	None
NetUseDel (local)	None
NetUseEnum (local)	None
NetUseGetInfo (local)	None

User

The NetUserPasswordSet function controls a user's password account on a server. NetUserPasswordSet can be executed only on a remote server running user-level security. Attempting to execute it on a local requester returns NERR_RemoteOnly.

Function	Differences in Use
NetUserAdd	None
NetUserDel	None
NetUserEnum	None
NetUserGetGroups	None
NetUserGetInfo	None

Function	Differences in Use
NetUserPasswordSet	None
NetUserSetInfo	None

Appendix F. IBM C/2 Sample Program

The following is a sample IBM C/2 program for the OS/2 LAN Server Version 1.2 application programming interface.

```
/*****************************************************************
/*                                                               *
/* FILE NAME SHARENUM.C                                          *
/*                                                               *
/* MODULE NAME= SHARENUM.C                                       *
/*                                                               *
/* DESCRIPTIVE NAME= C SAMPLE PROGRAM FOR THE LAN SERVER 1.2 API *
/*                                                               *
/*                                                               *
/*   COPYRIGHT:  XXXXXXXXX  (C) COPYRIGHT IBM CORP. 1989      *
/*               LICENSED MATERIAL - PROGRAM PROPERTY OF IBM      *
/*               ALL RIGHTS RESERVED                             *
/*                                                               *
/*                                                               *
/* NOTES=                                                        *
/*                                                               *
/
/*********************-END OF SPECIFICATIONS-***********************
 * SHARENUM.C
 *
 *
 * Invocation:  Invoke this program from the OS/2 full-screen
 *              command prompt by typing:
 *
 *              SHARENUM [\\SERVERNAME] LEVEL
 *
 *              Where SERVERNAME is the name of the remote server and LEVEL
 *              is the level of the share_info data structure to be returned.
 *              If SERVERNAME is not specified, the program will execute on
 *              the local server.  Valid values for LEVEL are 0, 1, and 2.
 *
 * Output:      The program outputs information using the printf function,
 *              depending on the level with which it is called:
 *              LEVEL = 0
 *                For each shared resource, the netname of the resource.
 *              LEVEL = 1
 *                For each shared resource:
 *                  the netname
 *                  the type
 *                  the remark
 *              LEVEL = 2
 *                For each shared resource:
 *                  the netname
 *                  the type
 *                  the remark
 *                  the maximum number of uses
 *                  the current number of uses
 *                  the directory path to the resource on the server
 *
 * Functions:   This program consists of two functions: main and InfoOut.
 *              The main function parses the command line, then calls
 *              NetShareEnum to find out how many shares are available.
```

```
 *             The actual number of shares is then used to calculate the
 *             buffer length, based on the level of information requested.
 *             NetShareEnum is then called a second time with this buffer.
 *             If no error is returned, main then calls InfoOut, passing
 *             the level, a pointer to the buffer, and the number of shares
 *             returned by the API.
 *             The InfoOut function uses the printf function to output the
 *             share information, based on the level.
 *
 *
 */

/*****************************
 *     Include Files          *
 *****************************/

#include <stdio.h>            /* Declares the printf() function  */
#include <stdlib.h>           /* Declares the atoi() function    */
#include <ctype.h>            /* Declares the isdigit() function */
#include <process.h>          /* Declares the exit() function    */
#include <os2.h>              /* Defines OS/2 constants and data structures */
#include <neterr.h>           /* Defines Network error codes     */
#include <netcons.h>          /* Defines Network constants      */
#include <shares.h>           /* Required by NetShareEnum        */
#include <access.h>           /* Required by NetShareEnum        */


/**  Forward Declaration of InfoOut **/
void InfoOut (int, char *, unsigned short);


/*****************************************
 *             main                      *
 *****************************************/

main(argc, argv)
int argc;
char *argv[];
{

  int                rc=0;                 /* API return code storage */
  char              *servername;           /* Pointer to ASCIIZ string */
  char              *buf;                  /* Pointer to a buffer area */
  short              level=0;                   /* Level of share info */
  unsigned short     buflen=0;                  /* Size of buffer area */
  unsigned short     entriesread=0;       /* Number of shares returned */
  unsigned short     totalentries=0;     /* Number of shares available */


/*************************************************************************/
/* Begin command line parsing */

   switch (argc)
   {
     case 1:                        /* Not enough arguments, return syntax */
      printf ("\nSyntax: sharenum [\\\\servername] level\n\n");
      printf ("Where servername is the name of the remote server\n");
```

```
 printf ("and level is a digit (0, 1, or 2) specifying the\n");
 printf ("level of share information.  If \\\\servername is not\n");
 printf ("specified, the call is executed locally.\n\n");
 exit(1);
break; /* case 1 */


case 2:                    /* Only 1 parameter is passed to sharenum; */
                           /* assume a null servername and test for a */
                           /* valid level.                            */


 if ( strlen(argv[1]) > 1 )              /* String must be of length 1 */
 {
    printf ("Level must be an integer between 0 and 2.\n");
    exit(1);
 }
 else                                    /* Test for a digit */
 {
    if ( isdigit (argv[1][0]) == 0 )
    {
       printf ("Level must be an integer between 0 and 2.\n");
       exit(1);
    }
    else
    {
       level = atoi(argv[1]);            /* argv[1] is a digit */
       servername = "\0";                /* Assign null servername */
    }

 } /* end else strlen */

break; /* case 2 */


case 3:                    /* Both servername and level are passed; */
                           /* test for a valid level.               */


 if ( strlen(argv[2]) > 1 )       /* Level string must be of length 1 */
 {
    printf ("Level must be an integer between 0 and 2.\n");
    exit(1);
 }
 else
 {
    if ( isdigit (argv[2][0]) == 0 )      /* Test for a digit */
     {
       printf ("Level must be an integer between 0 and 2.\n");
       exit(1);
     }
    else
     {
       level = atoi(argv[2]);             /* argv[2] is a digit */
       servername = argv[1];              /* argv[1] is the servername */

  /* Note: servername MUST BEGIN WITH \\ */
     }
```

```
    } /* end else strlen */

   break; /* case 3 */

 } /* end switch(argc) */

 if ( level > 2 )                          /* level must be 0, 1, or 2 */
 {
    printf ("Level must be an integer between 0 and 2.\n");
    exit(1);
 }

/* End command line parsing.                                          */
/**********************************************************************/



/* First call NetShareEnum to find out how many shares are available.
   The number of shares available will be given by the totalentries
   parameter.                                                         */

       buflen = 0;           /* Zero buffer length should cause error */


       rc = NetShareEnum ( (char far *) servername,
                           level,
                           (char far *) buf,
                           buflen,
                           (unsigned short far * ) &entriesread,
                           (unsigned short far * ) &totalentries);


       if ( rc == NERR_Success )              /* No shares on server */
       {
         printf ("\nServername: %s   ", servername);
         printf ("Level: %d   ", level);
         printf ("Return code from Enum: %d\n", rc);
         printf ("Entries read: %u   ", entriesread);
         printf ("Total entries: %u\n", totalentries);
         printf ("\nNo resources are shared.\n\n");
         exit (0);
       }
       else if ( rc != ERROR_MORE_DATA )     /* Some unexpected error */
       {
         printf ("\nServername: %s   ", servername);
         printf ("Level: %d   ", level);
         printf ("Return code from Enum: %d\n", rc);
         printf ("\nCall to NetShareEnum failed.\n\n");
         exit (2);
       }

 /* Now we know the total number of entries and can calculate buflen,
    based on the level, as follows:                                   */

 switch (level)
 {
  case 0:
```

```
            buflen = (sizeof(struct share_info_0)) * totalentries;
    break;

    case 1:
          buflen = ((sizeof(struct share_info_1))
                    + (MAXCOMMENTSZ + 1))
                    * totalentries;
    break;

    case 2:
          buflen = ((sizeof(struct share_info_2))
                    + (MAXCOMMENTSZ + 1)
                    + (PATHLEN + 1))
                    * totalentries;
    break;

    /* Level is not 0, 1, or 2. Print error message. */

    default:
      printf ("Level must be an integer between 0 and 2.\n");
      exit(1);
    break;

   } /* end switch(level) */

/* Allocate the buffer */

   buf = malloc(buflen);

/* A null pointer means insufficient memory */

   if ( buf == '\0' )
   {
     printf ("Insufficient memory to allocate share-info buffer.\n");
     exit(2);
   }

/* API call */
          rc = NetShareEnum ( (char far *) servername,
                              level,
                              (char far *) buf,
                              buflen,
                              (unsigned short far * ) &entriesread,
                              (unsigned short far * ) &totalentries);

           printf ("\nServername: %s   ", servername);
           printf ("Level: %d   ", level);
           printf ("Return code from Enum: %d\n", rc);
           printf ("Entries read: %u   ", entriesread);
           printf ("Total entries: %u\n\n", totalentries);

      /* If no error returned, call InfoOut to output the share info */
                if ( rc == NERR_Success )
                   InfoOut(level,buf,entriesread);

      /* Free the buffer */
                free(buf);
                exit(0);
```

```
 /* Note: There is a slight possibility that a share could have been added
          after the first API call such that totalentries would no longer
          reflect the real state of the server.  Thus, the buffer could
          be too small at the time of the second API call.  More error
          checking could be added to cover this case.                    */

} /* end main */


/************************************
 *           InfoOut                *
 ************************************/

void InfoOut(Level,Buf,EntriesRead)
int                Level;                    /* Level of share info */
char               *Buf;                    /* Pointer to a buffer area */
unsigned short     EntriesRead;             /* Number of shares returned */
{
int loop;
struct share_info_0  *BufPtr0;               /* Pointer to level 0 data */
struct share_info_1  *BufPtr1;               /* Pointer to level 1 data */
struct share_info_2  *BufPtr2;               /* Pointer to level 2 data */

  switch (Level)
   {
    case 0:                                 /* Output level 0 information */

      BufPtr0 = (struct share_info_0 *) Buf;          /* type cast */

      for (loop = 1; loop <= EntriesRead; ++loop)
       {

        printf("%s \n", BufPtr0->shi0_netname);       /* print the netname*/
        BufPtr0++;

       } /* end for */

      break;   /* case 0 */


    case 1:                                 /* Output level 1 information */

      BufPtr1 = (struct share_info_1 *) Buf;          /* type cast */

      for (loop = 1; loop <= EntriesRead; ++loop)
       {

    /* Print the netname, type, and remark */

        printf("Netname: %-12s ", BufPtr1->shi1_netname);
        printf("Type: %d \n", BufPtr1->shi1_type);
        printf("Remark: %s \n\n", BufPtr1->shi1_remark);
        BufPtr1++;
```

```
      } /* end for */

     break; /* case 1 */


   case 2:                                  /* Output level 2 information */

     BufPtr2 = (struct share_info_2 *) Buf;          /* type cast */

     for (loop = 1; loop <= EntriesRead; ++loop)
      {

   /* Print the netname, type, remark, maximum number of uses,
      current number of uses, and the path to the resource on
      the server.                                                  */

       printf("Netname: %s   ", BufPtr2->shi2_netname);
       printf("Type: %d \n", BufPtr2->shi2_type);
       printf("Remark: %s \n", BufPtr2->shi2_remark);
       printf("Max Uses: %u    ", BufPtr2->shi2_max_uses);
       printf("Current Uses: %u\n", BufPtr2->shi2_current_uses);
       printf("Local Path Name: %s \n\n", BufPtr2->shi2_path);
       BufPtr2++;

      } /* end for */

     break;   /* case 2 */


  } /* end switch */

}  /* end InfoOut */
```

Appendix G. PC LAN Program 1.3 Compatibility

The DOS LAN Requester with DOS provides several function calls that are not described in the main body of this book. These function calls are described here and are provided for compatibility with applications currently supported by the PC LAN program.

Function Call Overview

The following DOS Interrupt function calls are supported by the DOS LAN Requester.

INT	Function	Description
2AH	0000	Installation Check
2AH	0060	Network Print Stream Control
2AH	0300	Check Direct I/O
2AH	0400	Execute NETBIOS (Error Retry)
2AH	0401	Execute NETBIOS (No Error Retry)
2AH	0500	Get Network Resource Information
2AH	7802	Get user ID and logon status
2FH	B800	DOS LAN Requester installation check
2FH	B809	Network version check
2FH	B80F	Get Start parameter

Function Call Descriptions

The following pages contain descriptions of the function calls.

0000H (INT 2AH)
Installation Check

&pgrule.

Checks to see if the interrupt 2AH interface is installed

On Entry	Register Contents
AH	0

On Return	Register Contents

AH	Installed Flag
	`Zero ; Not installed`
	`Non-Zero ; Installed`

Remarks

A program can verify whether the interrupt 2AH interface is loaded by calling Interrupt 2AH Installation Check.

0060H (INT 2AH)
Network Print Stream Control

&pgrule.

Controls the truncation and concatenation of print streams for network printers.

On Entry	Register Contents
AH	06
AL	01 ; Set print output into ; concatenation mode 02 ; Set print output into ; truncation mode 03 ; Truncate print stream

On Return	Register Contents
AX	DOS error code if carry set. None if carry flag not set.

Remarks

DOS defines separate print streams for each of up to three redirected printers. These printers are LPT1 (or PRN), LPT2, and LPT3. These print streams are delimited by the following events:

- End of program
- Open and close of the files LPT1, LPT2, or LPT3
- Making a transition from printing with the IBM PC BIOS function INT 17H and printing with DOS (in both directions)
- Printing with INT 17H from different DOS processes (a process is created by the DOS EXEC function).

DOS is normally in Truncation mode, which means the stream delimiters listed above take effect. Concatenation mode (AL=01) causes DOS to ignore the stream delimiters. In this mode, the streams are delimited only when a DOS command returns to asking for user input. This is either when a single DOS command ends or when a DOS Batch (.BAT) ends. This allows output from several commands to be kept together if the commands are in a Batch file. Set Concatenation mode changes the stream state for all redirected printers. It has no affect on nonredirected printers.

Set Truncation mode (AL=02) returns DOS to Truncation mode. It is used to cancel the Set Concatenation mode function. This call does not truncate streams, the next stream delimiter will cause the truncation. Truncation mode is automatically set at the end of all DOS batch files and at the end of any command not in a batch file. Set Truncation mode changes the stream state for all redirected printers. It has no affect on nonredirected printers.

Not all programs print output in such a way that DOS can determine when a print stream ends. Printing using INT 17H is the most common occurance of this situation. Truncate Print Stream (AL=03) allows a program to indicate that the

data currently printed is a complete stream and it should be ended and printed. This function serves the same purpose as the Ctrl+Alt+PrtSc key that is available to users to end print streams. Truncate Print Stream truncates the streams for all redirected printers. It has no affect on nonredirected printers. The DOS LAN Requester program must be started for this function call to work.

0300H (INT 2AH)
Check Direct I/O

&pgrule.

Checks to see if an absolute disk access is allowed to the device.

On Entry	Register Contents
AX	0300H
DS:SI	Pointer to ASCIIZ disk device name

On Return	Register Contents
Carry flag	Set if absolute access is denied Clear if access is allowed

Remarks

The Check Direct I/O function call provides a check to see if a direct disk access is allowed to the specified device. Direct disk access is the use of DOS interrupts 25H and 26H or BIOS interrupt 13H. Use Check Direct I/O in programs that perform direct disk access. If the device is redirected or this function returns with carry set, then the program should not perform direct disk I/O.

Use Check Direct I/O to eliminate disk data integrity problems that can result from multiple concurrent processes updating the DOS disk data structures.

The device pointed to by DS:SI must include the colon (:). The path may be a full path or only the drive specifier.

Interrupt 2AH (Installation Check) should be done before the Check Direct I/O function call. If Interrupt 2A is not installed (AH=0), then absolute disk I/O is allowed. Programs should not use the Check Direct I/O function call frequently since it *may take some time to run.* If constant checks are needed, save the results of the first check, and check the saved result.

Note: The DOS LAN Requester must be loaded for the function call to execute properly.

0400H (INT 2AH)
Execute NETBIOS (Error Retry)

&pgrule.

Executes the specified NETBIOS function call with error retry support provided.

On Entry	Register Contents
AX	0400H
ES:BX	Pointer to the network control block (NCB)

On Return	Register Contents
AH=0, AL=0	No error
AH=1, AL=X	NCB error occurred. AL contains the error code X.

Remarks

Execute NETBIOS is reserved for use to support additional functions by capturing the intended function call and providing compatibility with extended functions. To ensure hardware independence, use the interrupt 2AH function. Do not use the interrupt 5CH function provided by the network adapter.

This function call provides error retry support for the following NETBIOS errors:

09H - No Resource Available
12H - Session Open Rejected
21H - Interface Busy.

The DOS LAN Requester retries the NETBIOS command that caused the error a number of times. To provide their own error recovery support, applications should invoke the Execute NETBIOS (0401H) function call.

On entry, the ES:BX register pair points to an NCB.

Note: The DOS LAN Requester must be loaded for the function call to execute properly.

0401H (INT 2AH)
Execute NETBIOS (No Error Retry)

&pgrule.

Executes the specified NETBIOS function call with *no* error retry support provided.

On Entry	Register Contents
AX	0401H
ES:BX	Pointer to the network control block (NCB)

On Return	Register Contents
AH=0, AL=0	No error
AH=1, AL=X	NCB error occurred. AL contains the error code X.

Remarks

Execute NETBIOS is reserved for supporting additional functions by capturing the intended function call and providing compatibility with extended functions. To ensure hardware independence, use the interrupt 2AH function call. Do not use the interrupt 5CH function provided by the network adapter.

This function call does not provide error retry support. (Execute NETBIOS function call 0400H provides error retry support.) Applications should provide their own error recovery if needed.

On entry, the ES:BX register pair points to an NCB.

Note: The DOS LAN Requester must be loaded for the function call to execute properly.

0500H (INT 2AH)
Get Network Resource Information

&pgrule.

Returns the number of local network names, network commands, and network sessions available to an application after the DOS LAN Requester has started.

On Entry	Register Contents
AX	0500H

On Return	Register Contents
AX	Reserved
BX	Number of network names available
CX	Number of network commands available
DX	Number of network sessions available

Remarks

The get network resource information allows an application to determine how many local network resources are available for use after the DOS LAN Requester starts. The application should invoke the function call before using any network commands. The application should maintain its own network resource count so that it does not exceed the values returned from get network resource count. Failure to comply with this requirement may cause unpredictable results from both the DOS LAN Requester and the network application.

7802H (INT 2AH)
Get User ID and Logon Status

&pgrule.

Provides the user ID and logon status information for the current user of the DOS LAN Requester program.

On Entry	Register Contents
AX	7802H
ES:DI	Address of 8 byte buffer

On Return	Register Contents
AL	Zero: No user logged on Non-zero: User logged on
ES:DI	Buffer contains user ID, ASCII, padded with blanks.

B800H (INT 2FH)
DOS LAN Requester Installation Check

&pgrule.

Checks to see if the DOS LAN Requester is installed. If the program is installed, Installation Check returns the installed network components.

On Entry	Register Contents
AX	B800H

On Return	Register Contents
AL	Network installed flag: Zero ; Network not installed Non-Zero ; Network is installed
BX	Installed component flag (bit flags): xxxxxxxx xxxx1xxx : Redirector xxxxxxxx 1xxxxxxx : Receiver

Note: The *x* represents an undefined value.

Assembler Usage

```
MOV   AX,B800H                ;AX - function code
INT   2FH                     ;Call function
CMP   AL,0                    ;Is network installed?
JE    NOT_INSTALLED           ;No, network not installed
TEST  BX,RECEIVER_FLAG        ;Is Receiver installed?
JNZ   RECEIVER_STARTED        ;Yes, Receiver is installed
TEST  BX,REDIRECTOR_FLAG      ;Is Redirector installed?
JNZ   REDIRECTOR_STARTED      ;Yes, Redirector is installed
```

Remarks

Use interrupt 2F (Installation Check) to determine if the DOS LAN Requester is installed. If the program is installed, Installation Check returns the components of the program that are installed. To determine which configuration the users specified when they started the network, perform the component checks in the order specified in the preceding assembler code. This order is required because a given configuration may install more than one component.

B809H (INT 2FH)
Network Version Check

&pgrule.

Returns the version level of DOS LAN Requester.

On Entry	Register Contents
AX	B809H

On Return	Register Contents
AH	Minor version number
AL	Major version number

Remarks

A network installation check (AX = B800H), should be performed first to determine if the network has been installed.

The Network Version Check function returns the hexadecimal equivalent of the DOS LAN Requester version.

B80FH (INT2FH)
Get Start Parameters

&pgrule.

Provides to the DOS application information on the start parameter values that are defined for the DOS LAN Requester.

On Entry	Register Contents
AX	B80FH
CX	Number of bytes to return
ES:DI	Address of output buffer

On Return	Register Contents
AX	Zero: Network started Non-zero: Network not started
CX	Number of bytes returned in buffer unchanged if network not started
ES:DI	Start parameter value in buffer (see remarks)

Remarks

The data representing the start parameter will be placed in the buffer in the following format:

```
DB   01H              Major Version
DB   00H              Minor Version
                      CONFIG Flags:  If bit on, then the value was specified
                                     when the network was started.
                      Bit 0 On = /NSV ≠ 0
                      Bit 1 On = /NMS ≠ 0
                      Bit 2 On = /API
                      Bit 3 On = /HIM
                      Bit 4 On = /LIM
                      Bit 5 On = /ENC
                      Bit 6 On = /POP
                      Bit 7 On = /EMS
                      Bit 8 On = /RPL
                      Bits 9 - 12 Reserved
                      Bit 13 On = RDR Started
                      Bit 14 On = RCV Started
                      Bit 15 On = User is currently logged on
DB  15 DUP(' ')       NET START Machine Name - blank padded
DB  0                 ASCIIZ ended
DB  9  DUP (0)        NET START Domain Name - ASCIIZ ended
DB  0                 Word Align
```

The following items are described in the description of the NET START command in the *DOS LAN Requester User's Guide.*:

```
DB  32 DUP (' ')        /WRK ASCII numeric - blank padded
DW  ?                   /SRV
DW  ?                   /ASG
DW  ?                   /NBC
DW  ?                   /NBS
DW  ?                   /BBC
DW  ?                   /BBS
DW  ?                   /PBC
DW  ?                   /PBS
DW  ?                   /PFS
DW  ?                   /PFT
DW  ?                   /PWT
DW  ?                   /KUC
DW  ?                   /KST
DW  ?                   /NVS
DW  ?                   /NMS
DW  ?                   /NDB
DW  ?                   /MBI
DB  ?                   NETBIOS Machine Name Number
DB  ?                   NETBIOS Group Name Number
DW  ?                   Sessions Required for Configuration
DW  ?                   Commands Required for Configuration
DW  ?                   Names Required for Configuration
DB  ?,':'               NET START Path (LANROOT)
DB  127 DUP('?')        ASCIIZ ended
```

Appendix H. LAN API Manifests

This appendix lists the manifests associated with variable-length ASCIIZ strings that are pointed to by data structure components used in the OS/2 LAN API. The items listed under the **Component** column are either offsets to, or pointers to variable length ASCIIZ strings. These ASCIIZ strings can be from 0 to some maximum number of bytes long in most instances. The maximum length for the variable-length ASCIIZ strings is a manifest, or constant, that is defined with a value in the NETCONS.H header file.

The variable length ASCIIZ string should not be greater in length than its manifest+1 bytes. The +1 is to allow for the ending NULL character of the string.

The **Manifest** column lists the manifests defined in NETCONS.H.

The **Component** column lists the data structure components that act as a pointer or are offset to a variable length ASCIIZ string.

The **Data Structure** column lists the data structure of which the component is a member.

The **Category** column lists the category to which the data structure belongs.

The information is as follows:

Manifest	Component	Data Structure	Category
CNLEN	computername	print_other_info	Alert
	computername	user_other_info	Alert
	ae_so_compname	ae_sesslogon	Auditing
	ae_sf_compname	ae_sesslogoff	Auditing
	ae_sp_compname	ae_sesspwerr	Auditing
	ae_ct_compname	ae_connstart	Auditing
	ae_cp_compname	ae_connstop	Auditing
	ae_cr_compname	ae_connrej	Auditing
	ae_ra_compname	ae_resaccess	Auditing
	ae_rr_compname	ae_resaccessrej	Auditing
	ae_cf_compname	ae_closefile	Auditing
	ae_am_compname	ae_aclmod	Auditing
	ae_um_compname	ae_uasmod	Auditing
	ae_no_compname	ae_netlogon	Auditing
	ae_nd_compname	ae_netlogondenied	Auditing
	ae_al_compname	ae_acclim	Auditing
	coni1_netname	connection_info_1	Connection
	wki0_computername	wksta_info_0	Requester
	wki0_logon_server	wksta_info_0	Requester
	wki1_computername	wksta_info_1	Requester
	wki1_logon_server	wksta_info_1	Requester
	wki10_computername	wksta_info_10	Requester
	sesi1_cname	session_info_1	Session
	sesi2_cname	session_info_2	Session
	sesi10_cname	session_info_10	Session

Manifest	Component	Data Structure	Category
	usri2_logon_server	user_info_2	User
	usri11_logon_server	user_info_11	User
	usrreq1_workstation	user_logon_req_1	User
	usrreq2_computer	user_logon_req_2	User
	usrlog2_computer	user_logon_info_2	User
DEVLEN	ch0_dev	chardev_info_0	Serial Device
	ch1_dev	chardev_info_1	Serial Device
	ui0_local	use_info_0	Use
	ui1_local	use_info_1	Use
DNLEN	wki0_langroup	wksta_info_0	Requester
	wki1_logon_domain	wksta_info_1	Requester
	wki10_langroup	wksta_info_10	Requester
	wki10_logon_domain	wksta_info_10	Requester
	usrlog1_domain	user_logon_info_1	User
MAXCOMMENTSZ	grpi1_comment	group_info_1	Group
	sv1_comment	server_info_1	Server
	sv2_comment	server_info_2	Server
	sv3_comment	server_info_3	Server
	shi1_remark	share_info_1	Share
	shi2_remark	share_info_2	Share
	usri1_comment	user_info_1	User
	usri2_comment	user_info_2	User
	usri10_comment	user_info_10	User
	usri11_comment	user_info_11	User
	usri2_full_name	user_info_2	User
	usri10_full_name	user_info_10	User
	usri11_full_name	user_info_11	User
	usri2_usr_comment	user_info_2	User
	usrlog2_full_name	user_logon_info_2	User
	usrlog2_usrcomment	user_logon_info_2	User
MAXDEVENTRIES * (DEVLEN)	cq1_devs	chardevQ_info_1	Serial Device
MAXWORKSTATIONS * (CNLEN)	usri2_workstations	user_info_2	User
	usri11_workstations	user_info_11	User
NNLEN	ae_ct_netname	ae_connstart	Auditing
	ae_cp_netname	ae_connstart	Auditing
	ae_cr_netname	ae_connrej	Auditing
	cq0_dev	chardevQ_info_0	Serial Device
	cq1_dev	chardevQ_info_1	Serial Device
PATHLEN	acc0_resource_name	access_info_0	Access
	acc1_resource_name	access_info_1	Access
	ae_ra_resname	ae_resaccess	Auditing
	ae_rr_resname	ae_resaccessrej	Auditing
	ae_cf_resname	ae_closefile	Auditing
	ae_am_resname	ae_aclmode	Auditing
	ae_um_resname	ae_uasmode	Auditing
	ae_al_resname	ae_acclim	Auditing
	fi1_pathname	file_info_1	File
	fi3_pathname	file_info_3	File

Manifest	Component	Data Structure	Category
	wki0_root	wksta_info_0	Requester
	wki1_root	wksta_info_1	Requester
	sv2_userpath	server_info_2	Server
	sv3_userpath	server_info_3	Server
	shi2_path	share_info_2	Share
	usri1_home_dir	user_info_1	User
	usri1_script_path	user_info_1	User
	usri2_home_dir	user_info_2	User
	usri2_script_path	user_info_2	User
	usri11_home_dir	user_info_11	User
	usri2_parms	user_info_2	User
	usri11_parms	user_info_1	User
	usrlog1_script_path	user_logon_info_1	User
UNLEN	username	print_other_info	Alert
	username	user_other_info	Alert
	ae_so_username	ae_sesslogon	Auditing
	ae_sf_username	ae_sesslogoff	Auditing
	ae_sp_username	ae_sesspwerr	Auditing
	ae_ct_username	ae_connstart	Auditing
	ae_cp_username	ae_connstop	Auditing
	ae_cr_username	ae_connrej	Auditing
	ae_ra_username	ae_resaccess	Auditing
	ae_rr_username	ae_resaccessrej	Auditing
	ae_cf_username	ae_closefile	Auditing
	ae_ss_username	ae_servicestat	Auditing
	ae_am_username	ae_aclmod	Auditing
	ae_um_username	ae_uasmod	Auditing
	ae_no_username	ae_netlogon	Auditing
	ae_nd_username	ae_netlogondenied	Auditing
	ae_al_username	ae_acclim	Auditing
	coni1_username	connection_info_1	Connection
	fi1_username	file_info_1	File
	fi3_username	file_info_3	File
	wki0_username	wksta_info_0	Requester
	wki1_username	wksta_info_1	Requester
	wki10_username	wksta_info_10	Requester
	sesi1_username	session_info_1	Session
	sesi2_username	session_info_2	Session
	sesi1_username	session_info_10	Session
RMLEN	ui0_remote	use_info_0	Use
	ui1_remote	use_info_1	Use
SNLEN	ae_ss_svcname	ae_servicestat	Auditing
WRKHEUR_COUNT	wki0_wrkheuristics	wksta_info_0	Requester
	sv2_srvheuristics	server_info_2	Server
	sv3_srvheuristics	server_info_3	Server

Glossary

This glossary contains terms specific to the LAN Server version 1.2 Application Programmer's Reference. Additional information can be found in the *IBM Dictionary of Computing*(SC20-1699).

access permission record. The information describing how users or groups can access the shared resource of a server. It contains the net name of the resource, user names and group names, and permissions granted for each user name and group name.

account. The record of a user on a server. A user must first have an account on a server to use any of the shared resources of the server.

admin. A function requirement specifying that the calling process must have administrative privileges to execute the function. This requirement is listed in parentheses after each function with the requirement.

alert. A notification of registered clients of a system event. OS/2 LAN Requester/Server provides a basic set of alerts. To add to them, use the Alert functions.

alert table. A list of registered users and groups of users, known as clients, to be notified each time a defined system event occurs. Clients can be registered as mailslots or semaphores.

Alerter service. See *service*.

anonymous pipe. A one-way data storage buffer maintained in RAM and used for interprocess communications (IPC). See *named pipe*.

API. See *application programming interface*.

application. A program or set of programs that perform a task; for example, a payroll application. For the OS/2 LAN Server, see *private application* and *public application*.

application programming interface (API). A formally-defined programming language interface between an IBM system control program or a licensed program and the user of a program.

ASCIIZ string. A null-ended (\0) ASCII string; a common string type used with the C programming language.

batch file. A file containing DOS commands organized for sequential processing while in DOS mode; files that are identified with a .BAT extension. For OS/2 mode, see *command file*.

broadcast message. A message sent to all users on the local area network (LAN).

centralized logon server. The server that verifies the logon password of a user name in a centralized logon security system.

client process. (1) A program that establishes a connection to (opens) a named pipe. (2) In Presentation Interface, a process that uses a service or dynamic link library. A process is a client of the service or library.

command. (1) A request from a terminal for performance of an operation or execution of a program.

command file. A file containing OS/2 commands organized for sequential processing while in OS/2 mode; files that have a .CMD file name extension. For DOS mode, see *batch file*.

computer. For LAN purposes, a computer is either a requester or a server. A computer can have only one computer name by which it is known to OS/2 LAN Requester/Server.

computer name. A name given a network requester or server. A computer name can be no longer than CNLEN bytes (as defined in the NETCONS.H include file). An example is:

```
workstal
```

communication device. A device connected to a serial communication port. When shared, a communication device is known by the name of the communication device queue to which it is connected. See also *device name*.

configuration. (1) The task of defining the devices, features, parameters, and programs for a system. (2) The arrangement and relationship of the components in a system or network.

connect. To redirect a local device name to a shared resource on a server.

connection. A direct communication link from a local device to a shared resource on a server.

continue. To restart a OS/2 LAN Requester/Server service or resource that was paused. See also *pause*.

device. In OS\2 LAN Server, a drive (for files resources) or port (for printers and serial devices) that is assigned when a resource is used. See *communication device*, *disk device*, and *device driver*.

device driver. The executable code needed to attach and use a device such as a display, printer, plotter, or communications adapter. See *device*

device name. A name assigned to identify a specific printer, disk drive, or other peripheral device. A device name can be no longer than DEVLEN bytes (as defined in NETCONS.H include file). Examples include:

```
G:
LPT5
```

directory. A structure for organizing files into groups. A directory can contain files and subdirectories of files in the format *\dir\...\filename.ext*

disk device. A device assigned a device name that stores information in the form of files.

distributed logon security. A security system in which each server verifies user permissions.

domain. A set of servers that allocates shared network resources within a single logical system.

event. A particular defined state of the network or of a service (such as when a disk drive reaches its complete disk capacity).

event name. A name of a particular type of system event, limited to EVLEN bytes (as defined in NETCONS.H). An example is:

```
TONEALERT
```

explicit connection. The establishing of a connection by means of the NetUseAdd function, where a local device name is redirected to the shared resource of a server.

file name. (1) The name used by a program to identify a file. (2) The portion of the identifying name that precedes the extension.

file handle. A binary value that represents an open file; used in all I/O operations.

group. A logical organization of users that have IDs according to activity or resource access authority.

group name. A name assigned to a particular set of user names. A group name can be no longer than GNLEN bytes (as defined in NETCONS.H). An example is:

```
businessgrp
```

handle. An arbitrary integer value that the OS/2 program returns to a process to represent a system resource so that the process can return the value to the OS/2 program on subsequent calls to use the resource.

implicit connection. The establishing of a connection to the shared resource of a server by specifying the UNC name of the resource.

IPC. See *interprocess communication.*

interprocess communication (IPC). The ability of local and remote processes to transfer data and messages among themselves; used to offer services to and receive services from other programs on the network.

LAN. See *local area network.*

LAN group. The set of computers to which a given computer belongs. A requester can belong to only one LAN group.

LAN path name. A computer name followed by one or more directory names, then followed by a file name (such as *\\print2\styles\info.zap*).

local computer. The computer where a user is working. See also *remote.*

local area network (LAN). (1) Two or more computing units connected for local resource sharing. (2) A network in which communications are limited to a moderate-sized geographic area such as a single office building, warehouse, or campus, and that do not extend accross public rights-of-way.

local device. A device physically attached to the local requester—that is, the drives in the computer and any machinery connected to its parts. This is a contrast to a remote device which the requester accesses with the LAN software.

local device name. A name by which a device is known (for example, C:\ or LPT1).

log. A file containing a historical list of information. With OS/2 LAN Requester/Server, several kinds of logs can be set up: an error log, statistical log, message log, and audit log.

log off. To remove a user name and password from a requester. See also *log on.*

log on. To enter a user name and password at a requester to enable access to the LAN. The user must log on at the beginning of each computer session. At the end of a work session, the user should log off. See also *log off.*

logon script. A set of commands executed when a user logs on to a requester.

logon security. A security and permissions system restricting who has access to the information, settings and devices of a LAN.

mailslot. A buffer that can store or forward messages to users or applications.

mailslot name. A name of a buffer that can receive messages. Mailslot names must be preceded by **\mailslot**, as follows:

```
\mailslot\name
```

message. A buffer or file of data sent to a messaging alias.

message forwarding. The ability to reroute messages intended for a user or application on one computer to a user or application on another computer.

message logging. The process of saving all incoming messages to a file.

messaging alias. A registered name that is used to receive messages.

messenger service. See *service.*

name. The label by which OS/2 LAN Requester/Server knows a user, a shared device and so forth. In the OS/2 LAN Requester/Server documents there are references to user name, computer name, net name, device name, etc. In addition, most system directories require a *pathname* parameter that identifies the drive, directory, subdirectories and specific name that the LAN must follow to get from the local requester to the device that is to be accessed.

named pipe. A data storage buffer that is maintained in RAM; used for interprocess communication (IPC). See *anonymous pipe.*

netname. A name assigned a shared resource with which remote users and processes establish connections. A net name consists of a computer name and the path name of where the resource is located. A net name can be no longer than RMLEN bytes (as defined in NETCONS.H). An example is:

```
\\server1\tools\excel
```

network. A configuration of data processing devices and software connected for information interchange.

network-aware application. An application that is implemented to use network resources to its benefit.

password. A word owned by a user or shared resource to prohibit other users from accessing a specified resource. User passwords can be no longer than PWLEN bytes; Share passwords can be no longer than SHPWLEN bytes (both parameters are defined in NETCONS.H). Examples include:

```
My_password
JLD*BDP
```

path. The path is the route that one device on a LAN takes to access a remote device. Path descriptions usually include a drive specification; a directory name (the root directory is specified with a backslash only (\)); and optional subdirectories, such as *\bin\src*).

path name. A file specification that describes the route from the current directory on the local requester to the file, in the following format:

```
devicename:\directory\subdirectory\...\filename
```

A path name can be no longer than PATHLEN bytes (as defined in the NETCONS.H include file). A backslash (\) precedes each directory name and file name. For example:

```
E:\BIN\USR\NETCONS.H
```

is the path name to the NETCONS.H file in the user subdirectory of the bin directory on the E drive. To specify a path name on the local drive, forgo the drive name, beginning the path with *\directory.*

pause. To suspend a OS/2 LAN Requester/Server service or function. See also *continue.*

permission. The permission setting on a shared resource determines which users can use the resource. Permission is also used to refer to the user's privileges. On the LAN, certain levels of permission can be set, giving the user various degrees of freedom in accessing devices and in reading or changing information. See *permission levels.*

permission levels. The degree to which a user can use a shared resource. Three permission levels (Guest, User, and Administrator) are provided in the OS/2 LAN Requester/Server software.

pipe. See *anonymous pipe, named pipe.*

pipe name. The name given a buffer that allows two processes to communicate serially with each other. A pipe name must be preceded by \pipe\, as follows:

```
\pipe\buffername
```

print device. A device that copies data from a computer onto paper. A print device is known to OS/2 LAN Requester/Server by the name of the port to which it is connected. See device names for more information specifying a print device in a LAN routine.

private application. An application maintained by an individual user and not available across a network. Contrast with *public application.*

public application. An application maintained by the network administrator and shared with users on a network. Contrast with *private application.*

queue. An orderly list of elements waiting to be processed. OS/2 LAN Requester/Server software supports serial device queues and spooler queues.

raise. To notify a user or application of a particular event.

raw read and write server message block (SMB) protocol. The protocol used when an incoming or outgoing packet contains only data and no SMB format is used.

remote. A term describing any server, requester, or resource that is not located on the local computer where a process is executing.

remote administration. Conducting administrative tasks such as sharing the resource of a server from a remote computer.

requester. The computer from which a user or application is accessing server resources.

resource. Any device, application, drive, or information on a server that can be accessed by a requester.

separator page. A sheet of paper automatically added between documents printed by way of a printer queue, sometimes containing such information as name, date and time of job.

serial device. Any hardware device that processes ASCII characters (such as a printer, modem, or FAX machine).

serial device queue. A pool of serial devices. With serial device queues, an application communicates directly to the device, instead of submitting a job to the queue (as with spooled devices).

server. A computer on a local area network that controls access to shared resources such as files, printers, and modems.

server name. A name of a particular network server. A server name can be no longer than CNLEN bytes (as defined in NETCONS.H). An example is:

```
server1
```

service. The programs that perform the primary functions of OS/2 LAN Requester/Server and related software. The OS/2 LAN Requester/Server services are:

Service	Purpose
replicator	Provides for file replication
dlrinst	Downloads code from servers to requesters
requester	Basic OS/2 LAN software
server	Software to perform administrative tasks such as sharing resources or assigning permissions and privileges
messenger	Software to send messages
netpopup	Software to receive incoming messages
alerter	Software to notify of an event
netrun	Software to remotely execute programs
netlogon	Software for centralized-logon server
pcdosrpl	Supports remote IPL of DOS requesters.

service name. A name of a network service. A service name can be no longer than SNLEN bytes (as defined in NETCONS.H). An example is:

```
NETPOPUP
```

session. A link between a requester and a server. A session is established the first time a requester requests to use the resource of a server.

share. To make a local resource available to remote users or other processes. Only local resources can be shared.

shared resource. The resource of a server that can be accessed by a requester on the network.

SMB. See *raw read and write server message block protocol.*

spool file data type name. A spool file data type defines the type of print jobs that a printer queue can process. A spool file data type name can be no longer than DTLEN bytes (as defined in NETCONS.H). An example is:

```
IBMQSTD
```

spooler. A program that intercepts the data going to a device driver and writes it to disk. The data is later printed or plotted when the required device is available. A spooler prevents output from different sources from being intermixed.

time stamp. A record of the time at which a system event occurred. Time stamps are figured on the basis of the LAN system clock as the number of seconds passed since January 1, 1970. Time stamps are used for statistics.

use. To establish a connection from a local device to a shared resource.

user name. The unique name assigned to each person granted access to a LAN system. To log onto a requester, the person must enter a user name and a self-assigned password (see also *password*). The system uses the user name to keep track of who is performing which operations.

A user name can be no longer than UNLEN bytes (as defined in NETCONS.H). An example is:

```
shannong
```

UNC name. See *Universal Naming Convention name.*

Universal Naming Convention (UNC) name. A name given to a device, computer, or resource under the UNC

to allow users and applications access to the resource across the network. An example is:

```
\\Server\drive\file.ext
```

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W