MICROSOFT®

50% OFF ORIGINAL PRICE

LAN MANAGER

A Programmer's Guide

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data

Ryan, Ralph, 1948-
The Microsoft LAN manager : a programmer's guide / Ralph Ryan.
p. cm.
ISBN 1-55615-166-7 : $29.95
1. OS/2 (Computer operating system) 2. Microsoft LAN manager (Computer program) I. Title.
QA76.76.063R93 1990
650'.028'55369--dc20

90-5418
CIP

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 FGFG 4 3 2 1 0

Distributed to the book trade in Canada by General Publishing Company, Ltd.

Distributed to the book trade outside the United States and Canada by Penguin Books Ltd.

Penguin Books Ltd., Harmondsworth, Middlesex, England
Penguin Books Australia Ltd., Ringwood, Victoria, Australia
Penguin Books N.Z. Ltd., 182-190 Wairau Road, Auckland 10, New Zealand

British Cataloging in Publication Data available

Project Editor: Megan E. Sheppard
Technical Editor: Michael Halvorson
Acquisitions Editor: Dean Holmes

(Second-class mailslots are one of the few parts of LAN Manager that do not use the Client-Server model.)

Because part of WKSTA.EXE is dedicated to second-class mail, the client workstation can receive as well as send second-class mail. This also allows a limited broadcast facility. For example, data written to a mailslot named *\MAILSLOT\WHO goes to all workstations in the domain with a mailslot named WHO. Mailslots are discussed in more detail in Chapter 10.

Error Logging Under OS/2, a third part of WKSTA.EXE is dedicated to error logging. Chapter 8 describes the error-logging APIs, which allow error information to be recorded in a file for later use. Software error conditions that cannot be directly reported to the user are noted in the error log.

However, if the redirector encounters a reportable error condition, it cannot use the error-log APIs: These APIs require information to be written to a file, and as a system-level program, the redirector does not have access to OS/2's file-system APIs. The redirector error information is therefore kept in a memory buffer, and the WKSTA.EXE thread (an application-level program with access to the OS/2 file system APIs) can periodically execute the appropriate error-log API calls. The WKSTA program is a LAN Manager service and thus can be started, stopped, paused, and continued.

Interstation Messages

Most workstations also run two services that provide interstation messages: the Messenger service and its companion, the Netpopup service.

The Messenger Service

Chapter 5 describes the Message APIs, which allow programs to send messages and to log received messages to a file. Messages are always sent to a name: It might be a username, or it might be an alias name. There are Message APIs for adding or deleting additional names by which messages might be received.

The Messenger service also provides message forwarding. This feature allows messages for a particular name to be received at another workstation.

The Netpopup service

Whenever a message arrives at the workstation, the Netpopup service is notified by the Messenger service. Netpopup then uses the *VioPopup* APIs to write to the screen that a message has arrived. It also displays part of the message text on the screen.

A few distinctions

In talking about the Messenger service, it is useful to make a distinction between two similar-sounding features: *messaging* and *electronic mail*.

- Messaging is a method of sending text from either a file or the keyboard to another workstation. Successful messaging requires that the destination be running the Messenger service and that it register the receiving username or aliasname. If either of these conditions is not met, the message is not sent.
- Electronic mail, or "e-mail," lets you send data between workstations. Electronic mail is always delivered: If the destination name isn't present, the message is stored on the network until it can be reliably delivered. Although e-mail isn't currently a part of core LAN Manager, some vendors have added their own e-mail service to LAN Manager.

Note also that messaging differs from mailslots: Mailslots are a LAN Manager API for interprocess communication; messaging is an inter-workstation service for sending text messages.

The User Interface

To most users, the user interface *is* LAN Manager: It is all they see and care about. To a programmer, the user interface is a simply a part of the greater LAN Manager architecture. The user-interface programs are C applications; they have no secret knowledge of the underlying system, and they use the same APIs discussed throughout this book. In fact, by using the LAN Manager APIs, programmers can create their own user interface that retains all functions of the original. In addition, some examples later in the book show how to extend the functions of the user interface.

The Server

LAN Manager servers run under OS/2 or UNIX. Figure 2-6 shows the basic architecture of a LAN Manager server. Note that a server is also a workstation under LAN Manager.

Figure 5-4. *continued*

```
rem directories to the specified network pathname.
rem
rem This file requires the NETCDX.EXE program and creates the
rem temporary file NETCDY.CMD in the current directory.

netcdx.exe %1 %2 > netcdy.cmd
call netcdy
del netcdy.cmd
```

The Message APIs

The Message APIs allow programs to interact with the Messenger service. Three sets of Message APIs are available:

- Name APIs work with message names, alias names, and name forwarding.
- Send APIs send memory buffers or files to specified names.
- Log file APIs work with a log of received messages.

Any user, regardless of privilege level, can issue a Message API locally. The key to understanding these APIs is understanding how the Messenger service works at the workstation and server levels.

The Messenger Service

One of the properties of the NetBIOS is the ability to add unique names to the network adapter card. The Messenger service allows messages to be received by any name that is on the adapter card. When the workstation is started, the computer name is added, and when the user logs on, the *NetWkstaSetUID* API adds the username. At this point, the Messenger service receives messages sent to either of these names (which could be the same). The Message APIs allow additional names, called *aliasnames*, to be added to the adapter card, and messages can be received by these names as well.

The Messenger service also allows names to be forwarded. (The Messenger service at each end has a copy of the name that is marked appropriately with "forwarded to" or "forwarded from" information.) Forwarded names can be unforwarded, which removes the remote copy of the name and restores delivery to the original computer.

NOTE: *The Message APIs take text strings that are case sensitive. The LAN Manager user interface uses the same APIs but always maps names to uppercase.*

Messenger service protocols

The following rules are enforced by the Message APIs:

- All Message APIs except *NetMessageBufferSend* and *NetMessageFileSend* require the Messenger service to be started at the workstation.
- If the Message APIs are called remotely, the Messenger service must be started at the remote computer, with the exception of *NetMessageBufferSend* and *NetMessageFileSend*.
- When messages are received at a workstation, they are buffered in a memory buffer whose size is controlled by the LANMAN.INI parameter *sizmessbuf*. If this buffer is too small, the message cannot be received.
- If logging is enabled (see *NetMessageLogFileSet*), the Messenger service writes the messages out to the log file.
- If the Netpopup service is started, the message is written to a pop-up screen at the receiving workstation.

Multiple networks

A workstation can maintain a NetBIOS driver for more than one network at a time. Chapter 12 shows how to configure a workstation for this. If a network is configured as a managed network, the workstation can send and receive messages on the network. If more than one managed network exists, the workstation can send and receive messages on all of them.

Using the Message APIs

To use the Message APIs in a program, you must include NETCONS.H and MESSAGE.H at compile time. *NetMessageBufferSend* and *NetMessageFileSend* must link with the NETOEM.LIB stub library. The rest of the Message APIs must link with the NETAPI.LIB stub library. With the exception of the MessageSend APIs, the Messenger service must be started for the API call to be successful. The MessageSend APIs require only that the Workstation service be running.

The message data structures

The *NetMessageNameEnum* and *NetMessageNameGetInfo* APIs make use of the *msg_info_0* and *msg_info_1* data structures. The *msg_info_0* structure contains the following member:

Member Name	*Description*
char msgi0_name[CNLEN+1]	A message name up to CNLEN characters in length and a 0 terminator.

The Workstation Service

The Workstation service is the primary LAN Manager service. Because it maintains information and tables used by many APIs, it must be installed (started) before any other services can be installed. After it has been installed, it can be paused, continued, or stopped:

- The workstation can be paused and continued. This causes all PrintQ and Comm redirection to be paused. This does not affect any open handles on redirected device names, but all other PrintQ and Comm connections revert to their local meanings. For example, if LPT1 is the local name for a connection to \\srv\laser, pausing the workstation makes LPT1 refer to the first printer device until the service is continued.
- If the workstation is paused or continued, the *arg* argument is a bitmask with the following meanings:

Bitmask	*Meaning*
SERVICE_CTRL_REDIR_PRINT	Pause PrintQ redirection
SERVICE_CTRL_REDIR_COMM	Pause Comm redirection

- The workstation can be stopped. However, if the server is installed, the workstation uninstall will fail.

The Message Service

The Message service lets workstations receive text messages from other workstations. With the exception of the *NetMessageBufferSend* and *NetMessageFileSend* APIs, all Message APIs require that the Message service be installed. Under OS/2 this service can be uninstalled, but it cannot be paused or continued.

The Netpopup Service

When the Message service indicates that a message has arrived, the Netpopup service uses the OS/2 Vio functions to print the message on the workstation console. Under OS/2 this service can be uninstalled, but it cannot be paused or continued.

The Server Service

The Server service is the primary service at the server. If the Server service is paused, no new sessions or new connections can be created, and no new opens are allowed. Existing sessions, connections, and opens are unaffected. The Server service can also be continued and uninstalled.

continued

Error Return	Meaning
ERROR_INVALID_PARAMETER	The *priority* or *class* argument contains an illegal value, or the message size is larger than the mailslot maximum message size.
ERROR_BROKEN_PIPE	The mailslot does not exist. It might have been deleted.
ERROR_BAD_NETNAME	A broadcast cannot be made unless the *class* argument is 2.

An overview:
The *DosWriteMailslot* API is used by client processes to write messages into the mailslot. The *name* argument must be a mailslot already created by a call to *DosMakeMailslot*. The mailslot can be created by another process which can even be running on another computer on the network. If the mailslot is local, the name is of the following form:

```
\MAILSLOT\mailname
```

If the mailslot is remote, the name is of the following form:

```
\\computername\MAILSLOT\mailname
```

A special form of the remote name allows a broadcast message to be sent. If the name is of the following form, the * indicates that the message is to be written to a mailslot on all computers in the domain (langroup) that have created a mailslot of this name:

```
\\*\MAILSLOT\mailname
```

You could also use the form

```
\\domain\MAILSLOT
```

to broadcast into the specified domain.

The broadcast can be done only for second-class mail.

The *priority* of a message ranges from 0, the lowest priority, to 9, the highest priority. Higher-priority messages will generally be delivered ahead of lower-priority ones, although this will not be true when the mailslot is already full.

There are two classes of messages, indicated by the *class* argument:

- If *class* is 1, the message is transported reliably and returns an error if it cannot be written into the mailslot. The *DosWriteMailslot* call will block until the write completes or until the failure condition is detected.
- If *class* is 2, there will be no indication if delivery fails.

First-class messages can be sent only to mailslots on LAN Manager servers. Second-class messages can be sent to mailslots on workstations and can be broadcast to the entire domain (langroup). Second-class mail is limited in size. LAN Manager version 2.0 allows a maximum of 444 bytes less the size of the full UNC name for the mailslot.

The *timeout* argument tells the *DosWriteMailslot* call how long to wait for room in the mailslot. The special value 0xFFFFFFFF means to wait indefinitely.

Modeling the Messenger Service

The LAN Manager Messenger service is implemented using a special SMB protocol and direct calls to NetBIOS. This gives it compatibility with older PC-LAN and MS-NET systems. You could also use mailslots to implement Messenger service functions, but this would not be interoperable with PC-LAN or MS-NET messaging.

The following programs are not a complete messaging service, but they do illustrate the use of mailslots for this purpose. MMESG.C (Figure 10-8) is a LAN Manager service program. If you add the statement

```
MMESG=service\mmesg.exe
```

to your LANMAN.INI file and put the executable file in the SERVICE directory of the LANMAN root directory, this new service can be started using the command

```
net start mmesg
```

The MSEND.C program (Figure 10-9) uses the MMESG service. It assumes that the first argument is a workstation name (or * for a domain broadcast) and sends the rest of the command line as a message. Messages are sent by constructing a remote mailslot name from the first argument and the name \MAILSLOT\EXAMPLE\MSG and then using *DosMailSlotWrite* to send a second-class message. A header on the front of the message contains the sender's name.