Changes & Enhancements not in the Book

Fido/FidoNet version 12R
7 November, 1989

Fido Software, Box 77731, San Francisco CA 94107
voice: 415-764-1688 data: 415-764-1629

New Commands

You will need to run the program "12R.EXE" to upgrade your COMMANDS.INI file before you can upgrade to Fido/FidoNet version 12R. It will add the new commands in the proper place plus add other version-identifying information. (It is safe to run the program more than once, and you can peek with your editor to see what it has done -- it's not a secret!)

Replying and Quoting

A new major feature has been added to Fido; "message quoting" when entering or replying to messages. The feature consists of a new Message Section command, a new command in the message editor, and enhancements to two existing message editor commands.

It works like this: while you are reading messages, you may find one that you want to reply to that has a number of points within it you want to address in your reply. Instead of taking notes, with the new W)rite-Buffer command (below) simply take a snapshot of the message. At any time before you disconnect, you can enter a new message, and read that snapshot into your message, where you can delete any parts that you don't want, which has been made a lot easier with the message editor enhancements.

Be warned: it is very easy to overuse message quoting, and generate huge and hard to read messages. A little goes a long way.

The edit buffer is a disk file named MSG.BUF; if the command line switch /I is used, the filename is modified in the same way as the log files.

New Message Section Command

A new command has been added to the Message Section: W)rite¡ Buffer. It saves a copy of the message you just read into the newly-coined "edit buffer". There is only one buffer, so only the most recent use of the W)rite-Buffer command is remembered. The contents of the Edit Buffer is cleared when a new caller connects; it is not preserved from one caller to the next.

The end result of this is that when replying to a message, you can include the original author's message as reference. The R)ead-Buffer command lets you "quote" each line with a ">" character to make quoted lines stand out.

For callers privilege 4 and above, the W)rite-Buffer command asks:

```
File to write to [CR=Buffer]:
```

Allowing you to save messages in a file you specify instead of the default buffer file. In this case, subsequent writes are appended to the file, letting you collect messages about a certain subject into a file.

Message Editor Enhancements

New command added: R)ead-Buffer. This reads the contents of the Edit Buffer into your message, as if you had typed it manually. It asks two questions:

```
Prefix each line with ">"? (y,N):
Force word-wrap on paragraphs? (y,N):
```

The first option lets you make the words of another person you are quoting stand out. The second is unfortunate, and is meant to help you compensate for messages generated by programs that do not properly support standard "word wrap" file format. First try it without it; if paragraphs look terrible (ie. a series of long lines followed by a short line, over and over) then delete the lines and try again with "force word-wrap" YES.

Because R)ead-Buffer reads in the entire edit buffer, you will need to delete the lines you don't want; see the D)elete-Line enhancement, below.

If you are privilege 4 or higher, R)ead-Buffer asks you an additional question:

```
File to read from [CR=Buffer]:
```

It will accept any valid pathname. The file better be a text file. Many uses come to mind -- canned answers for common questions, etc.

Besides the R)ead-Buffer command addition, two very old commands in the message editor have been radically improved.

D)elete-Line, which lets you delete a line from your message, now accepts a "range" of line numbers, with which you can delete many lines at once. Previously, if you wanted to delete (for example) lines 5 through 16, you had to enter "D 5" 12 times; now you can do it with a single "D 5-16" command.

I)nsert-Line was limited to inserting a single line in your existing text, too limited to be of much use. I)nsert-Line is now a true text-insert command; starting at the line number you specify, text is entered until you enter a blank line, as in normal text entry. Lines below your insertion point are "moved down" to make room for the inserted text.

New Change Sub-command

A new command has been added to the C)hange command menu; G)raphics. It is meant to allow the sysop to install an additional set of .BBS text files (WELCOME2,BBS, etc) that

contain graphic or other information, that can be chosen by each caller.

Alas, this feature isn't used yet -- hence the default privilege of 7 -- because I thought it out poorly and had to yank it at the last minute. Profuse apologies! I promise I will make it up to you soon!

New Main Section Command

T)riggers are manual controls over event execution. You can assign triggers to events defined in EVENTS.INI; you can put the same trigger on any number of events. Events without triggers are always on, just as they are in previous versions.

There are 8 triggers, which can have one of three possible settings: OFF, which means what it says; the trigger and any events that use it are disabled. ON allows that event to run when it's time comes. ONCE does also what it sounds like; after the event runs successfully, the trigger is turned OFF. This allows you to setup an event to run one time only, without having to remember to turn the event OFF after running it.

Triggers are placed on events when you define them in EVENTS.INI:

```
        All 9:00 360 FidoNet M          T=4
```

This event has trigger #4; that trigger must be ON or ONCE for that event to run when the time becomes 9:00AM. An example would be a special FidoNet event that sends mail to any system in the nodelist directly, for high priority mail, which you would enable with a trigger set to ONCE.

The T)riggers command lists events for you to help you see what you are doing.

Multi-Line Fido Installations

Once again, multi-line operation has been improved. There is now a way to run different modems on each of the different systems; See the new FIDO.INI option, system-path. Thanks go to Ken Ganshirt for this one.

It was never mentioned before, but Fido/FidoNet does file locking on CALLER.SYS, the caller file -- and does up to ten retries to

open it successfully. So you never have to worry about losing caller data.

For two-line Fidos, under DoubleDOS, DESQview, etc, you can now run both "sides" completely from within one "FIDO" subdirectory. Events and areas can be assigned to only one "task", by a new option in AREAS.INI and EVENTS.INI, to allow controlling which side executes what event, and additionally have message and file areas unique to a task.

Each systems task ID is the nnnn/I command line option; referring to the manual, the /I number is the "task ID" that makes each side unique, and makes Fido create unique logfiles, and handle message areas slightly differently.

For example, you have a two line Fido running, with sides numbered 1 and 2 (1/I and 2/I). The two systems are identical to the user, and you want to have only one side run FidoNet mail. The following example would do just that:

```
quick rush all  2:00 60   FidoNet A    ID=1
        all  0:00 1440  Page
```

The first event is assigned to side 1 only; under no circumstances will side 2 ever execute that event. The second event, a Page event, is shared; it has no ID number and therefore is common to both.

You should limit FidoNet events to only one side, when sharing a netmail area; all other event types can be shared without a problem.

Message and file areas can be assigned to each side in an identical manner. The other side will not be able to access the other sides areas. Areas without an ID= statement are shared by both sides.

```
msgarea= msg    D="Messages"    O=FidoNet ID=1
filearea= inbound U=outbound    D="Files" O=FidoNet ID=2
```

Options in AREAS.INI

Please note that previously, Fido let you specify options by the first character only; "O=F" was equivelant to "O=FidoNet", and so on. I hope you weren't too cheap with your typing -- as Fido now requires at least the first two characters now. This wasn't an arbitrary decision just to torment you -- the addition of the "O=Private" -- as opposed to the existing "O=Public" -- made this necessary. Sorry! SET-FIDO will inform you of your previous stinginess if necessary.

```
        O=Private
```

All messages entered in this area will be marked (PRIVATE). This helps those who run BBSs that may have marginal message contents; snoopy types simply cannot see anything, so you don't have to worry about getting caught.

O=Shared

Indicates to Fido that this message area is shared with another Fido or other program that can generate .MSG files in this directory -- this is meant to be used on multi-line Fido installations, to prevent message file contention. (It is actually file-locking, done at the right level for a change.) It causes Fido to recount messages whenever it needs to generate a new message file.

O=Anon

When a new message is created, it is marked From: Anon.

O=Public

Makes Fido not ask Private? (y,N); all messages are public.

New keywords in FIDO.INI

system-path <pathname>

Use this to define the pathname that Fido/FidoNet uses to locate the following files: CALLER.SYS, *.LOG, *.NDL, ROUTE.*, NODES.BBS. Normally Fido looks for these files in the default directory.

When running more than one Fido/FidoNet with a multitasker utility, you can now install each Fido/FidoNet in it's own subdirectory, but share critical files like the caller file, etc.

rings <1 - 255>

Fido/FidoNet normally answers the phone (well, modem...) on the first ring; this lets you change it to something else.

directory
key
aka
system
sysop
point
log
flag

These are DCM (Dutchie's Conference Mailer) keywords; Fido/FidoNet ignores them. DCM ignores Fido/FidoNet's keywords -- so you can use FIDO.INI to specify both Fido/FidoNet and DCM's installation, saving all that clutter and extra editing.

dot <1 - 32767>
alt-dot <1 - 32767>

These two define your Point address. The default is zero (no point address). Please see the section on POINTS below for details.

help-path <pathname>
bbs-path <pathname>
node-path <pathname>

IMPORTANT NOTE: There may be a change in the way Fido/FidoNet uses the help-path and bbs-path pathnames -- they may become "obsolete" to allow a decent implementation of the G)raphics command. The change will be no worse than simply using another keyword. It will be easy, and worth it.

These control from where Fido will access system files. help¡ path is where Fido will get all .HLP files. bbs-path is where Fido will look for all text .BBS files; quotes, WELCOMEs, BULLETIN.1 - BULLETIN.99, etc. Certain .BBS files remain are exempt: NODELIST.BBS, NODES.BBS, ROUTE.BBS, TIMELOG.BBS and of course all the FILES.BBS'. node-path is where Fido and FidoNet and the MAKELIST program will look for all of the NODELIST.* files.

SET-FIDO will not create these subdirectories for you; you must create them manually and copy the files into them. This is not necessary; it only allows you to keep a less cluttered FIDO directory.

zm-rx-type <number> ;DEFAULT: 0
zm-tx-type <number> ;DEFAULT: 0
zm-tx-start <number> ;DEFAULT: 1024

Please refer to the "ZMODEM" section in this errata sheet.

keep-nmp <YES,no>
wazoo <YES,no>
multi-tsync <YES,no>
fsc001 <yes,NO>
fsc011 <yes,NO>
system-name "Your System Name"
session-password <password>

Please refer to the "FIDONET" section in this errata sheet. The

default settings, if any, are shown above in CAPS -- unless you have a PARTICULAR REASON do not change the settings of these commands. They are for testing and protocol verification only.

multi-tasker <number>   ;DEFAULT: 0

An option to inform Fido of any "multitasker" program you might be using. Fido will run fine with any multitasker, even one not listed here; this is an option to potentially improve performance. You should see a slight performance increase. Replace <number> with one of the following: 0:Plain MSDOS; 1:DoubleDOS; 2:DESQview. Others may be defined later. (Please refer to the manual about the "nnnn/I" command line switch when running more than one Fido.)

external-login-A <number 3 - 255>
external-login-B <number 3 - 255>
external-login-C <number 3 - 255>
external-login-J <number 3 - 255>

This is part of a special option to allow Fido to run other login programs such as the uucp-to-FidoNet gateways software such as "UFGATE". (The unix uucp-to-FidoNet gateway software -- ask Tim Pozar at 1:125/555 for details.)

It enables a program or a person to login normally, but run another program instead of Fido. There can be up to 10 "external¡ logins" at one time. When properly installed, a caller that successfully passes the name and password section of the Fido

login exits Fido/FidoNet and runs a separate program, such as UFGATE. (It is up to the system operator to install the necessary programs and batch files to cause this to happen.)

You install this by first creating a batch file that runs your specified program via the DOS ERRORLEVEL convention. Then in FIDO.INI, you specify the external-login-A ERRORLEVEL to match ("A" can be any letter through "J"). This tells Fido that when a caller logs in with External-Login A, to exit to DOS with this ERRORLEVEL.

Next, for each program or person you wish to invoke the special login procedure, you assign a special attribute to an otherwise normal caller in the Fido caller file, "CALLER.SYS". This is done by setting the ADDRESS FIELD in the caller record to the exact string below:

        external-login A

Letters "A" through "J" identify which of the External-login definitions are used. The name and password fields are set normally. The address field is all the separates special logins from normal callers.

```
dial-prefix "string"
```

The string is prepended to the phone number from the nodelist files before dialing. A space is added between the prefix and the phone number. Suggestions: put "P" for pulse or private PBX access in there, instead of in NODELIST.BBS with XlatList and save a bunch of disk space and hassle.

```
NUMBER          PREFIX  RESULT
297-9145        (none)  ATDT2979145
297-9145        P..     ATDTP..2979145
642-1034        $DIAL   $DIAL 6421034
$dial_642-1034  (none)  $DIAL 6421034
$dial_642-1034  P..     P..$DIAL 6421034
```

Fido can execute script files instead of just dialing phone numbers. The script language is exactly the same as in FidoTerm, a shareware telecomm program available from the Fido Software BBS, except that the screen and console oriented commands have no effect or display on the screen.

A bucks character "$" in a phone number invokes the script processor. The text following the "$" is the script filename and arguments, and anything before the "$" is ignored. (That lets you mass-process phone numbers, using XlatList or "dial-prefix" in FIDO.INI without interfering.)

Arguments to script files must has spaces separating them; the usual "_" as defined for the nodelist file format is fine.

```
show-seen-by <yes,no>   ;DEF.: YES
```

This affects only users of echo-mail programs CONFMAIL and the like; if set to "NO", Fido suppresses the verbose "SEEN-BY" list of nodes.

```
quick-login <yes,no>    ;DEFAULT: NO
```

If "YES", the Q)uick-Login command at the local console logs in the first caller in the caller list (presumably the system operator). Very handy for local maintenance. Leave disabled if many people have physical access to the system.

```
modem-type 0    ;no modem
modem-type 2    ;Direct connection
modem-type 8    ;POPCOM 2400
modem-type 13   ;Multitech 224e
modem-type 12   ;see below
modem-type 21   ;Hayes V-series no ASB
modem-type 22   ;ditto, locked 9600
modem-type 23   ;ditto, locked 19,200
```

```
modem-type 24   ;USR HST, locked 38,400
modem-type 25   ;USR Courier 2400,
                ;Hardware Handshake
```

modem-type 0 prevents Fido/FidoNet from using the modem at all. In other words, Fido is usable only from the local console.

modem-type 2 is for direct-connect installations. When the CD line goes true, Fido assumes the online and connected state, at the baud rate set by maxbaud <number> in FIDO.INI. DTR is used to disconnect. You must use the new script facility to accomplish dialing. No modem initialization is done.

modem-type 8 is for the Prentice POPCOM 2400 baud modem.

modem-type 12 had a bug in version 12K; it issued USR Courier¡ specific commands, and many "generic" 2400 baud modems failed. This has been repaired; see modem type 25.

modem-type (14, 16, 17) have additional initialization commands: AT&H1&R2.

Zmodem file transfer protocol

This is a fully compatible, standard Zmodem implementation, with a few fancy features added. You can adjust Zmodems behavior with the two controls in FIDO.INI (details follow), because Zmodem can potentially accept data faster than your computer can handle. The default settings are quite conservative, and should work on all machines.

The block size used depends on the baud rate the connection is at, according to this table (see also zm-tx-start).

Block Size	Baud Rate
1024	over 2400
512	2400
256	1200 & below

Upon two consecutive errors on the same block, the block size is halved; minimum block size is 64 bytes. Upon twenty consecutive blocks with no errors and no line noise junk characters, block size is doubled; maximum block size is 1024 bytes.

Keep in mind the whole point of having high speed modems and protocols is so that you can run as fast as your machine allows; a modem capable of 1500 characters per second doesn't make your computer any faster, all it guarantees is that it won't hold you back anymore.

Now that a few months has gone by since ZMODEM was fist installed in Fido/FidoNet, I can offer more concrete advice.

Full streaming works in nearly all circumstances. The near-worst-case design is a 4.77MHz PC clone with an 80mS (slow!) hard disk, DOS 3.3, and an extremely fast modem, such as a US Robotics Dual Standard, locked at 38,400 baud. Even this combination is capable of doing 11,500 baud under good conditions. There is no need for even "AT" type hardware for high performance. (At least for file transfer speed alone.)

If you use a multitasker such as DoubleDOS, DESQview, Multilink, etc., and you experience high data error rates or lost data, then under these conditions please DO NOT USE Zmodem Receive full streaming. (See zm-rx-type.)

Zmodem Controls

The receive controls affect only how your Fido/FidoNet or FT program receives files; if someone else calls in to download files, Zmodem will go as fast as their Zmodem tells Fido or FT to go. (They may have done something like this on their end as well.)

REC'V: FULL STREAMING

FidoTerm:	ZRXTYPE 0 or 0/D
Fido:	zm-rx-type 0

When receiving, tells the sending program that it can accept data at maximum possible data rate, ie. full streaming. This is meant for machines that can accept data at "high speed", whatever that means to you.

REC'V: FULLY ACK'ED

FidoTerm:	ZRXTYPE 1 or 1/D
Fido:	zm-rx-type 1

When receiving files, every block will be acknowledged. (For sending, Fido/FT will do whatever the receiver says.) This is extremely conservative, and probably only needs to be used in extreme circumstances, such as under a heavily loaded multitasker.

TRANSMIT: VARIABLE WINDOW

FidoTerm:	ZTXTYPE 1 - 64/U
Fido:	zm-tx-type 1 - 64

The preferred method of defining a sliding window. When sending files, and the receiver says it can accept full streaming Fido/FT

will send data in full streaming mode, as long as it receives acknowledges from the receiver every so many [blocks]. The receiver sends occasional acknowledges, and the sender checks for them, without pausing the data flow. If the sender doesn't see an acknowledge it will stop and wait for one.

At 2400 baud and below, this has all the speed of full streaming, with improved error recovery. The slight penalty is the reverse channel does get used, which could slow some high-speed modems down.

Since the window size is stated in blocks, the size of the window depends on the baud rate and error rate; if many errors occur, Zmodem shrinks the block size, and hence the window shrinks too; if the error rate is exceptionally good, the block size increases as Zmodem increases block size. Higher baud rates start with larger blocks.

```
window size = [blocks] * block size
```

Try starting with [blocks] at 6, which works out to be a 1.5K byte window at 300 and 1200 baud, 3K at 2400, and 6K at 9600 and beyond.

HINT: Don't look at the Senders modem activity lights when adjusting window size; look only at the Receivers lights. The senders activity can be misleading; for example, the US Robotics HST has a 32K byte internal buffer, so Zmodem fills it quickly then sits and waits for window synchronization; don't let this fool you into thinking you could make it faster, you can't. Data can only flow out of the modem into the phone line as fast as it goes, all that increasing the window size will do is make error recovery slower.

TRANSMIT: FIXED SIZE WINDOW

```
FidoTerm:    ZTXTYPE 1024 - 65536
FidoTerm:    1024 - 65536/U
Fido:        zm-tx-type 1024 - 65536
```

This is the second method of defining a sliding window. It works the same as the previous method, except the size of the window is fixed, and specified in Kbytes. An 8K window is an 8K window, whether it contains 8 1024 byte blocks or 32 256 byte blocks.

TRANSMIT: START BLOCK SIZE

```
Fido:    zm-tx-start 64 - 1024
```

Normally Fido determines the data block size by baud rate and what the receiver can handle. On extremely bad phone lines, it may take too many errors to get the block size down to one that works; above 2400, where the block size starts at 1024 bytes, it

will take 16 errors (block size halved every four errors, four times) to get the 64 byte packet that may work best.

This option lets you specify the largest block size to start the transfer with. Try 128 or 64 if you have many errors due to phone line noise; if the connection is good, then after every 20 blocks the block size will double, and performance improve. 20 blocks doesn't take very long when the blocks are 64 bytes each!

This controls only the starting block size; the block size can still increase in the normal manner if there are no errors, as outlines in the beginning of this section.

Please make sure that you use only the following values: 64, 128, 256, 512, 1024.

Miscellaneous Additions

New system files: NODELIST.ZDX and NODELIST.NDX Fido can access any system listed in the nodelist with an average worst-case of four small disk reads -- performance with 10,000 nodes is much faster than most previous versions with only 500 nodes.

MAKELIST creates these, and Fido and all of the supplied tools use them. It is an additional index, and contains all the HOSTS and REGIONS and ZONES in the system. The existing NODELIST.IDX file has not changed in format nor use; external programs that use it are not affected.

WELCOME3.BBS, WELCOME4.BBS and WELCOME5.BBS are displayed right after the point where it now displays WELCOME2.BBS.

Incompletely uploaded or received files (carrier loss, Control-X abort, timeout, etc) no longer clutter the directories; Fido kills 'em.

New option at the More[c,Y,n] prompt: C == Continuous, ie. suspend the "more" function until the next prompt for input.

NODELIST.SYS is not used anymore. MAKELIST.EXE used to generate it, and FIDO.EXE read it. Fido now uses NODELIST.BBS directly.

embedded in .BBS text files such as WELCOME1.BBS, it performs certain special functions (Page 26 in the manual). The following were added:

Control-D Fido immediately displays a "More" pause prompt.

Control-X Suspends auto-line wrap until the next CR is found. This allows embedding long ANSI sequences without word wrap messing you up. (Remember mechanical typewriters? This is a "margin release"!)

Control-Z Fido treats this as the end of the file.

MsgMgr.EXE options added: New keyword that can be used within

MsgMgr script files:

LOGFILE logfilename

Normally MsgMgr logs it's activity in FIDO.LOG, the standard Fido log file. With this command, you can route log activity to any file or pathname or device, or to eliminate it entirely, to the device "NUL".

You can now specify the name of the script file that the message manager is to use, instead of just the default "MSGMGR.INI". For example, you could renumber/purge only your FidoNet area right before mail time, and then use MsgMgr in the usual manner after FidoNet.

MsgMgr will also now translate "LASTREAD" and "HW.DAT" files if they exist in each message area.

FidoNet

Once again, the FidoNet portion of Fido/FidoNet has received a major overhaul. At this point (12q) performance, simplicity, and compatibility should be just about "all there".

With version 12Q comes a rather radical simplification of overall FidoNet operation. Gone are all the confusing FidoNet event-type options. Since there are people looking at this who haven't seen a Fido/FidoNet since version 11, I'll sum up the changes here:

- True three-level addressing (v12)
- Continuous incoming mail (v12)
- Wazoo/Zmodem protocol (v12m)
- Usable continuous outgoing mail (v12m)
- File Requesting (v12m)
- True continuous outgoing mail (v12q)
- Incremental packeting (v12q)
- Basic point support
- Scheduled control of File Requests (v12q)

During this revision (12q), most if not all of the FidoNet-program implementors were working towards making their program adhere to the basic FSC001 protocol standard, and we all tested against each others programs as well. Yes, you can even file-attach to SEAdogs.

FidoNet, from the sysop's installation and operation point of view, was kind of a jumble of complex options in EVENTS.INI and less than satisfactory when trying to run continuous outgoing mail -- ie. to have packets ready for pickup at any time. I think it is safe to say that all of these problems have been fixed. You no longer need to have a zillion events throughout the day, and

newly entered messages no longer sit around until one of those zillion events comes by.

The FidoNet event types you specify in EVENTS.INI were completely revamped. The previous method involved complex and obscure

options that confused even me -- it was poorly thought out. There are now only three types of FidoNet event (described below). There is what I think a good sample installation that should cover most peoples needs described below. It should be easy to install and understand.

FidoNet Events

There are three types of FidoNet events, each described below.

Normal FidoNet

```
ALL  2:00  60  FIDONET A
```

This is the old standby FidoNet event type. It runs until it's time is up. Human callers are not allowed into Fido; it accepts only FidoNet mail.

Rush FidoNet

```
RUSH  2:00  60 FIDONET A
```

Very similar to the previous "vanilla" FidoNet event, except that when there is no more mail to send, ie. all the packets have been delivered or the maximum number or tries has been reached, the event terminates early.

RUSH FIDONET events are especially useful when combined with a T)rigger; you can define an event to run all day long (0:00 1440), and use it to manually override normal scheduling.

Continuous FidoNet

```
CONT  2:00  60  FIDONET A
```

True continuous outgoing mail. This causes Fido/FidoNet to make packets, and have them available for pickup or delivery at any time, while allowing human callers to access Fido freely.

When there is no human caller occupying Fido, and there are packets to deliver (according to route language files) FidoNet will make calls once per dial-interval.

Other FidoNet systems can call in at any time (well, assuming it's not in use), deliver FidoNet mail, and pick up packets

addressed to it.

If a human or a FidoNet mailer generates a message in the FidoNet message area, FidoNet will immediately add it to an existing packet or create a new one; and deliver the packet as per normal FidoNet routing controls.

QUICK FidoNet Option: Obsolete

The QUICK option is no longer used -- though SET-FIDO will not (for now) complain. All FidoNet events are now, by definition,

"QUICK". This means that you must run SET-FIDO if you change any of your ROUTE.* files. Which was strongly reccommended anyways. So now it's mandatory.

A Sample Installation

The following is a very good starting point for a full featured Fido/FidoNet installation for a node in the amateur FidoNet network. It is described in detail below:

```
RUSH    0:00    1440    FIDONET R       T=1
        2:00      60    FIDONET A
CONT    2:00    1380    FIDONET L
```

(Fido executes events by scanning the list of scheduled events from top to bottom, and runs the first event it finds that is runnable. The RUSH FIDONET event will only run (and therefore override the events that follow) when Trigger 1 is turned on.)

And here's a sample ROUTE.BBS file to go with this:

```
IF-SCHEDULE R   ;manual override
    SEND-ONLY
    SEND-TO, NO-ROUTE MYZONE

IF-SCHEDULE A   ;'ZoneMailHour'
    SEND-TO ALL

IF-SCHEDULE L   ;daytime mail --
    SEND-ONLY, DIAL-TRIES 1
    ;generate packets for all
    SEND-TO, NO-ROUTE MYZONE
    ;but call only my own net
    HOLD ALL NOT MYNET

END-IF
```

The first event, RUSH FIDONET R, is controlled by Trigger 1, and we'll assume usually turned OFF. (When OFF, the event is inert

and will not run.) When turned ON, FidoNet will repacket according to the route file; in this example, it will make packets for messages within our own zone (SEND-TO MYZONE), without host routing (NO-ROUTE MYZONE), and dial otu to deliver those packets as fast as possible (SEND-ONLY). It will stop when (1) if there were no messages to deliver, (2) as soon as all messages are delivered, or (3) the maximum number of tries is reached.

The second event, FIDONET A, is the normal, mandatory, FidoNet "ZMH". SEND-TO ALL simply means enable mailing to all nodes in the nodelist (note that for inter-zone messages, the contents of ROUTE.DEF (elsewhere...) will route messages to the proper zonegate). This event will run until completion; in this example, from 2:00AM til 3:00AM.

The third event, CONT FIDONET L, is the "background", continuous FidoNet event. It will run whenever the previous two are not. It will make packets according to the route language for tag L: packets only to your zone (SEND-TO MYZONE), no default host routing (NO-ROUTE MYZONE). FidoNet will make phone calls only to nodes in your own net -- HOLD ALL NOT MYNET.

Assume now that it's in the afternoon, and CONT FIDONET L is running. You are, for example, 1:125/0, the host for net 125. 1:161/12345 calls and delivers a packet with a message destined for 1:125/7. If there were messages for 1:161/12345 it would be on HOLD -- it is outside your net (HOLD ALL NOT MYNET) -- and it could be picked up at this time.

After it disconnects, Fido unpackets the message. FidoNet then discovers the new message for 1:125/7 -- it immediately creates a new packet for 1:125/7, and since that is within your own net, immediately calls it and delivers the packet. If '7 were busy, then Fido would run and wait for a caller. While Fido remains idle (no one calls in), every dial-interval FidoNet will run, and attempt to deliver that packet to '7.

And further: assume a human caller connects now, with that packet to '7 still undelivered, and goes into the FidoNet netmail area, and enters some messages: another to 1:125/7, one to 1:161/12345, and another to say 2:500/5. When the caller disconnects, FidoNet will packet those messages: the first will go into the existing packet for 1:125/7, the second to (say) a new packet for 1:161/12345, and the third will not be packeted at all -- you said SEND-TO, NO-ROUTE MYZONE so it just sits.

With the packets then updated, FidoNet runs again. It calls '7, connects, and delivers the packet. (The messages and packet can then be deleted.) The packet for 1:161/12345 is not delivered, since it is outside your net. FidoNet relinquishes control to Fido, allowing human or FidoNet callers in.

Points

Fido/FidoNet now (12N) includes basic Point addressing support. Later versions will provide complete "boss node" functions, so that Fido/FidoNet will be able to perform any possible FidoNet mailer function; zone gate, zone host, net/region host, ordinary node, point boss, point node.

Fido/FidoNet properly handles all point-addressed messages; it will scan for TOPT and FMPT Kludge lines, and incorporate them into the message address display.

When reading existing messages, Fido locates the TOPT and FMPT IFNA Kludge lines, like it always has for INTL lines, and incorporates them into the displayed address. To the user or sysop, there's nothing to even think about.

When entering a message, node address entry is as it always was, but you can add ".<1 - 32767>" to the node number, or just the dot followed by the point number, to send to points within your

own point network. For example: as 125/111, entering .33 would address a message to 1:125/111.33, etc. Same syntax and behavior as default net, etc.

When replying, Fido does the same as it does with nodes; the message is To: the From: node, unless it is the same as "us" in which case it reverses the addresses.

Internally, Fido generates TOPT and FMPT lines as the second and third lines in the .MSG file; INTL still comes first. Fido will locate any of these three lines as long as they occur within the first 256 bytes of text following the message header.

Note that Fido will display point numbers only if the point number is NOT zero. 1:125/111.555 will display that way; .0 will display as 1:125/111 only.

Wazoo Protocol

Fido/FidoNet now supports two network protocols automatically. One is "FidoNet", known in some circles as "FSC001", after the filename of the standards document generated by Randy Bush. Fido has supported this protocol since 1985.

The other, newer protocol is called "Wazoo", and was originally implemented in Wynn Wagner's Opus program, and now it seems the most popular program supporting Wazoo is "BinkleyTerm". Both also

do FSC001 style FidoNet. Wazoo operates in a similar manner, but uses Zmodem for it's transfers and can support "file requests", where the call originator can "download" files from the remote computer without the intervention of an operator to do "file attaches". (With appropriate controls, etc.)

There is no impact on existing Fido/FidoNet installations regarding security, setup or installation, etc. The choice of Wazoo vs. FidoNet is made automatically, though the system operator can make changes. The defaults will work fine in all cases.

There are FIDO.INI options that were added to control things. Most of the options should be left as-is.

```
fsc011 <yes,no>      ;default: NO
wazoo <yes,no>       ;def.: YES
multi-tsync <yes,no> ;def.: YES
fsc001 <yes,no>      ;default: NO
```

These are for special purposes only, mainly for verifying FidoNet FSC001 protocol compatibility. Unfortunately in the real world there are varying levels of compliance to FSC001; the default is pretty "loose", for maximum compatibility. fsc011 yes lets Fido/FidoNet accept so-called "DIETIFNA" protocol, which means it can skip the slow MODEM7 filename; however SEAdog does not accept TELINK blocks and file attaches will then fail. wazoo no forces Fido to do only FSC001, ie. pre-12M compatibility. multi-tsync no forces Fido back to 11W style single TSYNC character; there is

nearly no reason to do this. fsc001 yes makes FidoNet and it's XMODEM protocol driver conform to letter-of-the-law FSC001 specifications; alas, not many FidoNet "compatible" systems will then work, including many Fido/FidoNet programs!

system-name is an optional 60 character string that is the "name" of your system. Fido transmits this name to the remote computer during Wazoo sessions. session-password may be required for connecting to some other Wazoo-based systems; please make arrangements with the person requiring it.

File Requests

Fido now supports file request in either FSC001-type or Wazoo FidoNet sessions. A file request is a file transfer originated by the calling system, that requests one of more files by name; the called system then transmits the requested files in that same session.

The receiving system has full control over what it will and will

not allow to be requested; this is to prevent the obvious "file request *.*" getting copyrighted programs, critical data files (caller lists anyone?) and other problems.

There are shareware and free programs that will automatically generate a file request, given only the desired filenames and the node address to request from. What follows is the technical description of how it works.

A file request consists of a file with a special name sent to the receiving system, the one that will possibly honor the request. The filename is:

XXXXYYYY.REQ

Where: XXXX is the receiving system's net number, in four-digit hexadecimal, and YYYY the receiving system's four-digit net ! 00010002.REQ; a request to 125/111 would be 007D006F.REQ.

Inside this file, one per line, are the file(s) to be requested. Each line ends with a CR or CR/LF. Filenames can be any length, and may not contain pathnames or drive letters. (Fido will ignore them.) Filenames can include wildcards.

Generating File Requests

You can request files from within Fido; it is an option in the message editor in the FidoNet message area. You can enter any number of filenames, with wildcards, as will fit on the line. Fido will automatically generate the awful .REQ file for you.

Controlling File Requests

The receiver has a file called "FILEREQ.INI", that is the list of files that may be requestable from the system. Initially only two

sample files are requestable (see below), and the system operator needs to add to this list all files that you wish to be requestable remotely.

As provided by Fido Software, there are only two requestable files: "AB! is "about", and "FILES", which is the list of files requestable from your system. In the hobbyist FidoNet network, it is traditional (and useful!) to have these two files requestable at all times, so that other system operators can find out about your BBS without having to manually call and ask.

You can also restrict File Requests to specific hours, with the event type FILEREQUEST, in EVENTS.INI:

ALL 3:00 1380 FILEREQUEST

This allows file requests at any time except between 2:00 and

3:00AM, the Zone Mail Hour. A file request made while it is disabled will send the contents of the file "NOFREQ.BBS" to the requesting system as file XXXXYYYY.FRQ, where XXXXYYY is the requesting node's address, as described under the .REQ file. Presumably you'd list in NOFILEREQ.BBS the reasons the file request was not honored.

The FILEREQ.INI File

This file is used to control how Fido/FidoNet handles file requests. It! maintain, that contains the list of files and directories you wish to have available for other systems to file-request with their FidoNet type mailer.

You can also put comments into this file; comments are any line that begins with a semicolon, like so:

```
;
;Lines beginning with a ";" are comments,
;and are ignored by Fido.
;
```

Comments do however slow down processing, so try to keep them short.

All other lines in the file define requestable directories or specific files. The most popular method is to make the contents of a directory requestable. This is easy; simply list the directories you wish to make available, one per line.

```
C:/LISTS
C:/FIDO/TEXTFILE
C:/SHAREWARE
```

And so on. It means that all files within each directory are requestable. There is one odd side effect; if the filename you request exists in more than one directory, Fido/FidoNet will send you
every single one. Too bad. For example, "FILES.BBS". Fido will transmit! that contain it. Too bad if that's not what you MEANT, that's what you SAID.

Note also that "FILE" can contain wildcards; "*.*" for instance. It will of course return every single requestable file stored in your system. (Ugh.)

It is also useful to have "logical" file requests; for example, you want to announce that requesting the file "NODELIST" will always return the very latest nodelist file, no matter what it is really named, without having to constantly rename the file:

```
        NODELIST=LISTS/NODELIST.*
```

Whenever someone requests the filename "NODELIST", (the name to the left of the "=" equals sign) Fido will send them your file (after the "=" equals sign), that matches "LISTS/NODELIST.*". This lets files be requested by their logical names, no matter where they may reside on your disk. (For revenge, you can have a file called "CALLER.SYS" that returned something nasty instead of the a!

You can use this in other ways: you could respond to a request for "ALL-LISTS" with all the files you have, for example:

```
ALL-LISTS=LISTS/*.*
```

There can also be more than one entry with the same requestable name. For example, if you wanted to have a "kit" of files for new system operators requestable, you could convert the request for "NEW-SYSOP" to send many files:

```
NEW-SYSOP=LIST/NODELIST
NEW-SYSOP=NEWNODE.TXT
NEW-SYSOP=/TEXT/COORD.LST
etc
```

Fido will search the entire FILEREQ.INI file, and send all files that match all requests.

The third method is for when you want to make single files in arbitrary subdirectories available. For example, you might want to make certain files in your "/FIDO" directory available, but still maintain absolute security. Entering "FILENAME=FILENAME" works, but is tedious and redundant. There is a short form for when you just want to make a file requestable with it's original name, and not necessarily provide multiple files, etc:

```
C:/LIST/NODELIST.099
```

A file request for the filename portion (here, "NODELIST.099") causes your system to send that file, period. The pathname is used locally, in your system only; it is not requestable nor accessible. This shorthand lets you generate lists of files and place them, as-is, into FILEREQ.INI.

Nodemaps

Fido/FidoNet saves nodemaps it creates for each FidoNet schedule tag. (Nodemaps are what the Router produces by reading the ROUTE.* routing language files, and applying them to the file NODELIST.NMP.) There is one nodemap file (NODEMAP.tag) per

schedule tag, and each file is four bytes per node in the nodelist -- or 24K per schedule you use for a 6,000 node nodelist. FidoNet generates a nodemap the first time each event runs -- after that changing FidoNet schedules is nearly instantaneous. (You can disable this (why?!) with the FIDO.INI

command keep-nodemaps no.)

Routing Language Additions

```
    Zone 1                   ;current ZONE is 1
    BEGIN
      Zone 4 ZoneGate a:b/c  ;change ZONE to 4,
    END
                             ;zone is now 1 again
```

BEGIN and END add block structure to the router commands that change default behavior, mainly NET and ZONE. BEGIN saves the current state, and END restores them. NET and ZONE changes within a BEGIN/END group are local to that group; after the END, the values of NET and ZONE are restored to what they were at the time the BEGIN statement was executed. You can nest BEGIN and END up to four levels deep.

The one-argument commands POLL, PICKUP, NO-ROUTE, ACCEPT-FROM, HOLD, SEND-TO can be "stacked" together, for faster execution. For example, if you do a lot of things like:

        Send-To All, PickUp All

They can now be stacked onto one argument, as in:

        Send-To, Pickup All

The advantage is that all of the stacked commands are executed at once (when the "All" is read) instead of one at a time. "ALL" makes the router apply the commands to all nodes in the nodelist; stacking in this example is twice as fast.

        ALIAS-AS <alias>, <nodes...>

A powerful new route language command ALIAS-AS: Similar to the current ROUTE-TO command, you can force all messages routed to the alias node, regardless of other routing or files attached.

For example, you exchange mail with a person who runs a node with more than one alias address; for example 105/6, 105/0, 1/2 and 1/3 are all the same machine. You simply set (for example) 105/6 as the alias for the other nodes.

Please note that this is not another "route-to" command; while it does a "route-to" as part of it's action, it is a new command entirely. Route-to only does one level of indirection; alias-as adds a second. Alias-as can be thought of as a kind of "route-to route-to".

Route-to defines to what destination node a message goes to,

possibly (probably) not the one the message is addressed to.

Alias-as defines to whose packet those routed messages go into.

Route file processing

A new ROUTE file is introduced: ROUTE.DEF. Fido looks for and processes ROUTE.DEF before looking for ROUTE.(tag) and/or ROUTE.BBS. This lets you do routing controls in common with all FidoNet events.

New keywords were added to the route language processor:

```
IF-SCHEDULE <tag>
END-IF
```

Ken G's suggestion roundly applauded by everyone else. Even I now agree it is an excellent idea. It does what you think it does. You now have an alternative to all those scroungy little ROUTE.<tag> files; put them all into ROUTE.BBS (maybe keep ROUTE.DEF, it partitions that nicely and doesn't slow things down) for ease in maintenance.

If no IF-SCHEDULE statements are used, then FidoNet processes the route list normally; all statements are read and processed.

```
IF-SCHEDULE A
        schedule A statements ...
END-IF
```

If the currently executing schedule is "A", then the statements between the IF...END are executed, otherwise they are ignored.

```
IF-SCHEDULE M
        schedule M statements ...

IF-SCHEDULE B
        schedule B statements ...

IF-SCHEDULE D
        schedule D statements ...

END-IF
```

Not a big deal to figure out. Each IF-SCHEDULE ends the one before it (if any). Processing is as you'd expect.

If you have commands to execute for all schedules (say, PICKUP ALL) just place them before any IF-SCHEDULE statements.

```
        zone x  ZONEGATE node
```

This tells FidoNet to route all mail for Zone X to the specified node; the supplied ROUTE.DEF file implements IFNA type zone gating. There is no restriction on Fido's ZoneGate. For example, in ROUTE.DEF:

```
;
```

```
;Do IFNA Kludge type Zone Gating
;
Zone 2 ZoneGate 1:1/2
Zone 3 ZoneGate 1:1/3
```

```
        DIAL-TRIES n
        CONNECT-TRIES n
```

Maximum number of times to dial each node. (Default is "dial-tries" and "connect-tries", respectively, in FIDO.INI)

```
        MYZONE
```

A modifier keyword, meaning All nodes in my own zone.

```
        THISZONE
```

Another modifier keyword, meaning All nodes in the currently specified zone.