MakeNL -- Version 2.50

NODELIST Generator Program

by Ben Baker

Table of Contents

1. Introduction 1
   1.1 How to use this manual 2

2. MakeNL Operation 3
   2.1 Operational overview 3
   2.2 Operation at a Hub or simple Network 4
   2.3 Operation at a large Network 6
   2.4 Operation by a Regional Coordinator 10
   2.5 Operation by a Zone Coordinator 10
   2.6 Putting together a Distribution Nodelist Segment 13

3. Definitions 14
   3.1 Composite Nodelist 14
   3.2 Generic File Name 14
   3.3 Explicit File Name 14
   3.4 Archive File 14
   3.5 Difference File 15
   3.6 Submittal File 15
   3.7 Distribution File 15
   3.8 Absolute Directory Path 15
   3.9 Relative Directory Path 15
   3.10 Publication Date 15

4. MakeNL Command Line 17
   4.1 Control file name 17
   4.2 /TEST switch 17
   4.3 /PROCESS switch 18
   4.4 /MERGE switch 18
   4.5 /NAME=<network_name> 18
   4.6 Error return codes 19

5. MakeNL Directories 20
   5.1 Master Directory 20
   5.2 Update Directory 20
   5.3 Mailfiles Directory 20
   5.4 Upload Directory 20
   5.5 Other Directories 20

6. MakeNL Control File 21
   6.1 Control statements 21
      6.1.1 MAKe <segment_type> [<number> [<source_file>]] 21
      6.1.2 NAMe <network_name> 22
      6.1.3 PUBlish <day_of_week> 22
      6.1.4 PROcess <day_of_week> 23

6.1.5 MERge [<nodelist_name>] 24

6.1.6 PRIvate <disposition> 25
6.1.7 MINphone <minimum_parts> 25
6.1.8 BAUdrate <valid_baud_rates> 26
6.2 Defining nodelist segment directories 27
6.2.1 MASter <directory_path_name> 27
6.2.2 UPLoads <directory_path_name> 27
6.2.3 MAIlfiles <directory_path_name> 28
6.2.4 UPDate <directory_path_name> 28
6.2.5 BADfiles <directory_path_name> 28
6.3 Defining output 29
6.3.1 OUTFile <filename> 29
6.3.2 OUTPath <directory_path_name> 29
6.3.3 THReshold <arc_size> [<diff_size>] 29
6.3.4 ARC <n> 30
6.3.5 OUTDiff <generic_filename> 30
6.3.6 CLEanup 32
6.4 External message transmission 32
6.4.1 NETaddress [<zone>:]<net>/<node> 33
6.4.2 MESsages <directory_path_name> 33
6.4.3 SUBmit <address> [<flags>] 33
6.4.4 NOTify <type> [<flags>] 34
6.5 Defining special files 34
6.5.1 Output comments files 34
6.5.1.1 COPyright <explicit_filename> 35
6.5.1.2 PROlog <explicit_filename> 36
6.5.1.3 EPIlog <explicit_filename> 36
6.5.2 Comments from received files 36
6.5.2.1 COMments <explicit_filename> 36
6.6 DATa 36
6.7 FILes 37

7. Disk and Memory Space Considerations 38

8. Contents of MakeNL Distribution Package 40

9. MakeNL License Information 41

Table of Figures

Figure 1. Data Flow - Test Mode vs. Process Mode 3

Figure 2. Sample control files for Hubs and small Nets 5

Figure 3. Sample control file for large Nets 7

Figure 4. Sample control file for Regional Coordinator 9

Figure 5. Sample control file for Zone Coordinators 11

Figure 6, Sample control file for distribution lists 12

1. Introduction

FidoNet, a trade mark of Tom Jennings, was the name of the first amateur electronic mail program. There are now several mail server programs that support FidoNet protocols. Many amateur networks, large and small, are using one or more of these mail servers to communicate regularly.

The glue that holds an amateur network together is its nodelist, a document that describes the nodes in the network, and the relationship between them. To be effective, this document must use a consistent format, so that programs may process its information automatically. The FidoNet Technical Standards Committee maintains, among others, a document called FTS-0005.TXT that defines the format for a FidoNet compatible nodelist. A copy of that document is included in this distribution.

Furthermore, a nodelist must be accurate and current. An out-of-date phone number, for example, is useless. It must be updated regularly, and most networks publish a new nodelist each week. This is an enormous task in any network of more than a few dozen nodes. No one person can do it. But division of labor adds complexity.

Obviously, we need a tool that can help verify consistency, maximize accuracy and currency, minimize effort, and minimize cost. MakeNL is such a tool.

MakeNL supports the multilevel nodelist generation techniques presently in use in several FidoNet compatible electronic mail networks. It helps maintain nodelist segments at any level from Hub through Network, Region and Zone to the composite distribution nodelist.

MakeNL's main purpose for being is to create a "submittal file" (see definitions - section 3 on page 14) containing your nodelist update, and to send it to your coordinator. It can do this automatically, if you set up its control file properly. It is a complex program and looks intimidating, but it is not really difficult.

MakeNL has features to help in nodelist preparation at any level, but it is also adaptable. Features you don't need you may safely ignore, and MakeNL will not complain.

1.1 How to use this manual

Section 2 is a tutorial on setting up and operating **MakeNL**. It starts with the simplest case -- the lowest level of nodelist maintenance -- and expands in steps to Zone Coordinator.

Sections 3 through 8 contain detailed reference information. Section 2 necessarily refers to information contained in later sections. The early paragraphs of section 2 contain frequent forward references to assist the reader.

I recommend you read the entire document through once. Pay particular attention to those parts of section 2 that pertain to you, but at least skim the rest. Familiarize yourself with the layout and content of the various sections, but do not read for thorough understanding on your first pass.

Carefully study section 2 a second time. Refer to other sections for clarification of terms as necessary.

Select the sample control file that best represents your particular needs. Edit it to fit your particular environment, as indicated in section 2 and the comments in the control file.

Make copies of your master data files in a safe place (floppy diskette, or a directory tucked away in a far corner of your hard disk).

Familiarize yourself with **MakeNL** by actually running it, in both test and process mode. During this phase, I suggest you comment out any "Submit" or "Notify" statements in your control file to avoid generating spurious messages.

Introduce data errors to see how **MakeNL** handles them. Make sure you're comfortable with it before you put it on line.

When you're satisfied, delete all "test case" files. Copy your master files to your master directory. Make the necessary changes to enable your mail server's "external event" for **MakeNL** and you're up.

MakeNL simplifies your life. It looks imposing, but it's really not very difficult.

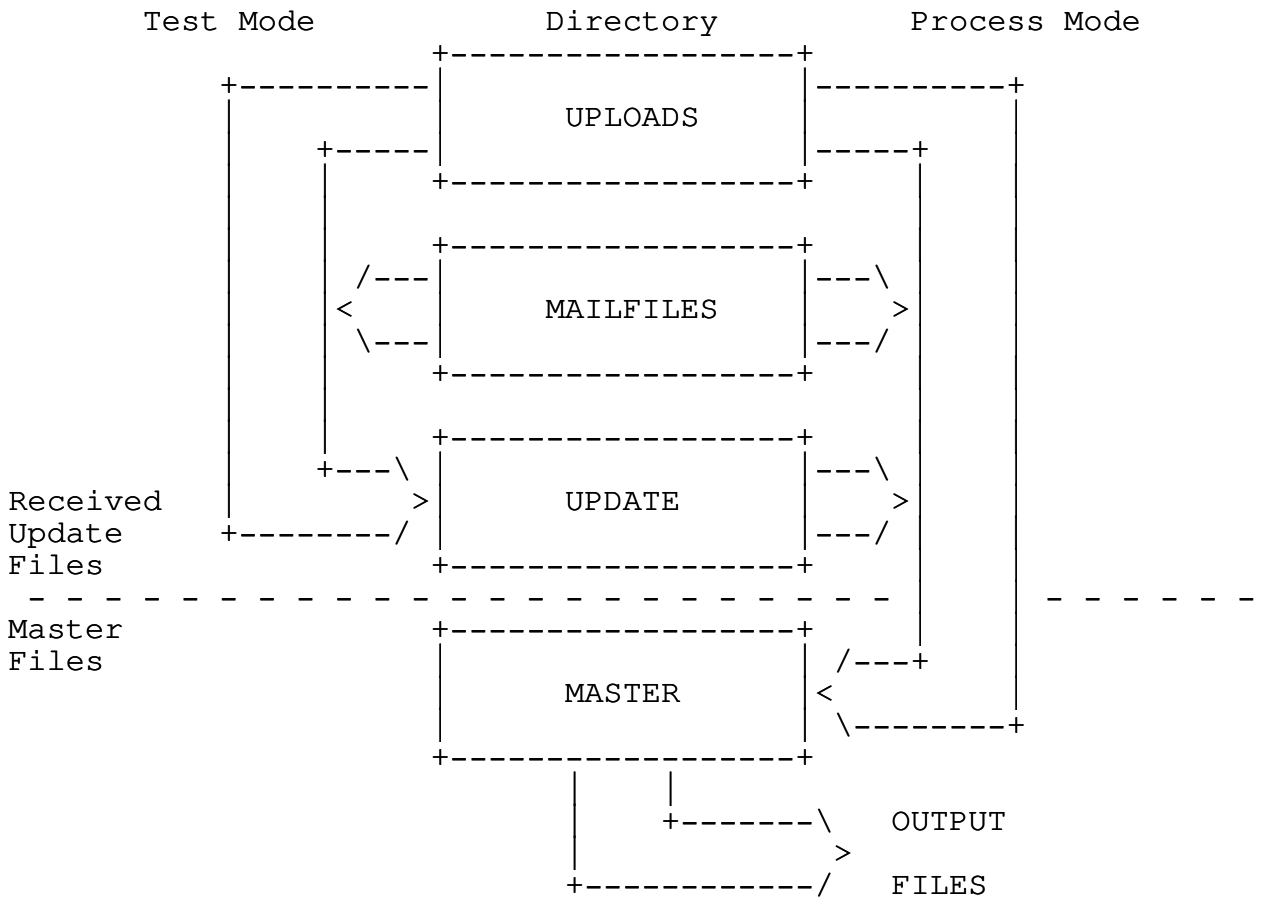

Figure 1. Data Flow - Test Mode vs. Process Mode

2. MakeNL Operation

2.1 Operational overview

MakeNL operates in two distinct modes, test mode and process mode. The mode of operation is controlled by the "PROcess" statement in your control file (see section 6.1.4 on page 23), or by the "/T" and "/P" command line switches (see sections 4.2 and 4.3 on page 17).

In test mode your personal nodelist segment, the part of the nodelist you maintain, and any update segments you have received from others are tested for errors. Submitters of any received files are notified as indicated by the NOTify statements in the control file, and their files are moved to the UPDate directory. (See section 5 on page 20 for a description of **MakeNL**'s direc-

tories.)

In process mode your MASter directory is updated by any files received since the last time MakeNL was executed in process mode. Master data files are used to produce your submittal and/or distribution files.

In general, MakeNL should be run in process mode once and only once each week. Ideally, this will be done by an "external event" from your mail server. It is then automatic, and one less thing you have to worry about.

It may be run in test mode as often as desired to test received files or changes in your own data.

MakeNL uses several directories to store data. Section 5 describes them in detail. Figure 1 illustrates the data flow from directory to directory in the two modes of operation. Above the dashed line, the data flow all deals with files received from others -- perhaps from your Hubs if you are a Network Host, or from Network Hosts if you are a Regional Coordinator. It is clear from this picture that if you do not receive nodelist updates from others (and many of you don't), you have no use for three of MakeNL's data directories, and a good deal of MakeNL's complexity disappears.

The following paragraphs give recommended set-up and operating procedures for various types of coordinators.

2.2 Operation at a Hub or simple Network

Here is where the nodelist update process begins. You have a small nodelist segment that you maintain. You do not receive any update files from anyone else. If you are a Network Coordinator and your network has Hubs, YOU maintain their data -- they just forward the mail. Yours is the easiest task, so your set-up for MakeNL is the simplest for you.

Figure 2 shows two variations of a MakeNL control file tailored to your needs. You may maintain your nodelist segment as a separate file, or you may embed your data in your MakeNL control file. (I prefer the latter method, but it is a matter of taste.) If you wish to maintain your data in a separate file, refer to the top half of Figure 2 (a facsimile of file HUB.CTL in your distribution package). If you want to keep your data in your control file, refer to the bottom half of Figure 2 (a facsimile of file NET-S.CTL in your distribution package). Refer to sections 5 and 6 as you study these files.

In either case, some minor editing of the control file you chose as indicated by comments in the file will have you nearly ready to go. When you have edited the control file to your liking, rename it MAKENL.CTL and move it to your mail server's home directory.

```
; This variation uses a "master data file" for input and
; sends output to the same file name.

make hub <100> <hub.100>  ; note explicit name of file
                          ; simplifies processing
outfile   <hub.100>       ; master and output files have SAME
                          ; name

submit   <host's address> ; where you send updates, CRASH
                          ; and/or HOLD optional
netaddress <your address> ; your network address here
messages   <directory>    ; path name to mail server's
                          ; mail area

; No other options are needed.  Master directory and output
; directory will default to the current directory.
```

```
; This variation uses "internal" data at the bottom of the
; control file.

make network <nbr>        ; No "master data file"  input data
                          ; is below.
outfile <explicit name>   ; output file

submit     <RC's address> ; where you send updates, CRASH
                          ; and/or HOLD optional
netaddress <your address> ; your network address here
messages   <directory>    ; path name to mail server's
                          ; mail area

; No other options are needed.  Master directory and output
; directory will default to the current directory.

data

; Your source data goes here.  The advantage of this over
; the version above is that MAKENL will not reformat or change
; your source data, even if you have errors.
```

Figure 2. Sample control files for Hubs and small Nets

Since you don't have many files to keep track of, you will execute **MakeNL** from your mail server's home directory, and let your MASter directory default to this directory as well.

Set up an external event in your mail server to execute once a week at a day and time early enough to meet the deadline set by your coordinator, but not much earlier than that. The batch file

fragment corresponding to this external event will execute the statement:

```
MAKENL
```

You will normally ignore MakeNL. It will take care of itself. When a change occurs in your data, edit your file using your favorite editor. After editing, execute the following DOS command:

```
MAKENL /T
```

just to be sure there are no errors. If MakeNL reports an error, repeat the above steps until you've got it right.

If you have added new nodes to your list, and wish to inform them of their node number assignments immediately, you need to get your changes into your mail server's nodelist. MakeNL can perform the first step for you if you execute the DOS command:

```
MAKENL /T /M[[=<path>\]<filename>]
```

The /M switch tells MakeNL to merge your data with an existing distribution nodelist into a temporary nodelist with an extension of ".999". To do this, MakeNL must find a distribution list not more than two weeks old. If /M is used by itself, MakeNL will look for NODELIST.htu (see section 3.2 on page 14 for a description of this notation) in the current directory. If either the directory or the stem generic name are different, they must be given as a value in the /M switch.

Examples

```
MAKENL /T /M                ; use NODELIST in current
MAKENL /T /M=\BBS\NODELIST ; use NODELIST in \BBS\
MAKENL /T /M=R10-LIST       ; use R10-LIST in current
```

When the external event triggers MakeNL in process mode (you are in bed, sound asleep) It will create a new copy of your master file, compare it to the old copy, and if nothing has changed, it will simply delete the new copy. If the file has changed, it will update your master file and create a submittal message to send it to your coordinator.

That's all there is to it! MakeNL will take care of getting your nodelist changes in on time if and when they occur.

2.3 Operation at a large Network

Your life is a little more complicated than above. You assemble your submittal segment from your own file plus files received from your Hubs. Not only do you need to figure out this section. You need to understand the forgoing one too, so that you may render assistance to your Hubs if necessary. But cheer up, it's not

too bad!

```
make network <nbr>          ; No "master data file"  input data
                            ; is below.

outfile <explicit name>     ; use generic name if output file
                            ; is very large

process    Wednesday        ; Set this day to meet deadline RC
                            ; gives you

master     <directory>      ; where master files reside
                            ; (default - current)
update     <directory>      ; where to save received files
                            ; 'till processing
mailfiles  <directory>      ; where mail server places received
                            ; files
uploads    <directory>      ; where BBS places uploaded files
                            ; (if needed)
badfiles   <directory>      ; optional "waste can" for files
                            ; with fatal errors

notify  receipt             ; notify both receipt and errors
                            ; CRASH and/or HOLD optional
submit     <RC's address>   ; where you send updates, CRASH
                            ; and/or HOLD optional
netaddress <your address>   ; your network address here
messages   <directory>      ; path name to mail server's
                            ; mail area

; No other options are needed.

data

; Your source data goes here.

files

Hub <nbr> <file name>       ; either generic or explicit file
                            ; name, but must match convention
Hub <nbr> <file name>       ; used by Hub.
Hub <nbr> <file name> <notification address>; if not "net/hub"
Hub <nbr> <file name>
;      etc.
```

Figure 3. Sample control file for large Nets

Refer to the figure 3 (a facsimile of file NET-L.CTL in your distribution package).

The submittal file for most local networks will be less than five kilobytes in size. For files this small, there is no advantage in using a generic output file name (see section 3.2 on page 14). Your Regional Coordinator will probably tell you to use an explicit file name. There is certainly no reason for you to require your Hubs to use generic submittal file names. Notice that the OUTFile statement in figure 3 indicates the use of an explicit name.

Rename NET-L.CTL to MAKENL.CTL and move it to the directory you wish to use as the current directory when you execute **MakeNL**. Usually, this will be your mail server's home directory. Edit the file as indicated in the comments in NET-L.CTL. If you only have two or three hubs, you may wish to comment out the MASter directory statement and let it default to the current directory.

Copy your own source data, the part you maintain which contains the "Host" statement into the control file between the DATa and FILes statements. Following the FILes statement, edit the HUB definition statements to match the Hubs who submit update files to you. Make sure the directories exist for all path names, that they are all **distinct** directories, and if you used relative path names, that they are relative to the directory from which you will execute **MakeNL**. Make sure the process statement names the right day so that you can meet your RC's submittal deadline. That should get your control file in order.

Move your current copies of your Hubs' segment files to the directory defined as MASter in your control file.

Set up an external event in your mail server to execute daily at a time early enough that, on your process day, you will meet the deadline set by your Regional Coordinator, but not much earlier than that. The batch file fragment corresponding to this external event will execute the statement:

```
MAKENL
```

or if you want to keep a log of activity

```
MAKENL    >>MAKENL.LOG
```

MakeNL is now set up to pretty well take care of itself. On non-process days the external event will execute **MakeNL** in test mode. It will scan the UPLoads and MAIlfiles directories for received files, test them for errors, send out notifications and move the files to the UPDate directory.

On the process day **MakeNL** will be executed in process mode. It will scan the UPLoads and MAIlfiles directories for late files, the UPDate directory for files received earlier in the week, and the MASter directory for files not updated this week.

```
make region <nbr>          ; No "master data file"  input data
                           ; is below.

outfile <generic name>     ; use generic name for output file

process    Thursday        ; Set this day to meet deadline ZC
                           ; gives you

arc 5      ; force ARC 6 or 7 to generate level 5 archives
           ; DO NOT use ARC5 with ARC versions earlier than 6

master     <directory>     ; where master files reside
                           ; (default - current)
update     <directory>     ; where to save received files
                           ; 'till processing
mailfiles  <directory>     ; where mail server places received
                           ; files
uploads    <directory>     ; where BBS places uploaded files
                           ; (if needed)
badfiles   <directory>     ; optional "waste can" for files
                           ; with fatal errors

notify  errors CRASH       ; notify of errors (CRASH and/or
                           ; HOLD optional)
notify  receipt            ; use either of these or both with
                           ; different attributes.

submit     <ZC's address>  ; where you send updates, CRASH
                           ;  and/or HOLD optional
netaddress <your address>  ; your network address here
messages   <directory>     ; path name to mail server's
                           ; mail area

data

; Your region source data goes here.

files

Net <nbr> <file name>      ; either generic or explicit file
Net <nbr> <file name>      ; name, but must match convention
                           ; used by Net.
Net <nbr> <file name> <notification address> ; if not "net/0"
Net <nbr> <file name>
```

Figure 4. Sample control file for Regional Coordinator

All files will be tested (again) for errors, and sent to your master output file. Your master copies of your Hubs' files will be updated as necessary.

If (and only if) this week's output is changed from last week's, your submittal file will be created in the current directory, and a "file attach" message will be generated to send it to your Regional Coordinator. All untouched by human hands!

When the data changes in your own nodelist segment, edit your data file (usually your control file) and execute the DOS command:

```
MAKENL /T
   or
MAKENL /T /M   ; see discussion in the previous section
```

Given a chance, **MakeNL** will do the right things at the right times. You worry about other things, like "Why can't Hub 999 send me a file without errors? I gotta get him to use MakeNL!"

2.4 Operation by a Regional Coordinator

For the most part, section 2.3 applies to you too, but there are some differences.

Figure 4 is a facsimile of REGION.CTL, the prototype control file you will use.

You have enough files you will want to get them out of the clutter of your mail server's home directory. You will set up a separate MASter directory (see section 5.1 on page 20).

Your output file is probably large enough that you will want to take advantage of the benefits of using a generic output file name. In fact, your ZC may require it.

If your region file is larger than 10 kilobytes, **MakeNL** will create a compressed archive by executing the program ARC. If you have version 6 or 7 of ARC and your ZC does not, you may want to force ARC to produce level-5 compatible archives.

Other than that, substitute Network for Hub, Region for Network and Zone for Region, and follow the procedures in 2.3 using the file REGION.CTL from your distribution package.

2.5 Operation by a Zone Coordinator

Life gets still more complicated, but is still manageable.

Preparing a zone segment is much the same as preparing the region segments of which it is made. Refer to sections 2.3 and 2.4 above, figure 5 (a facsimile of the file ZONE.CTL from your dis-

tribution package).

```
make zone <nbr>            ; No "master data file"  input data
                           ; is below.
outfile <explicit name>    ; use generic name for output file
                           ; if very large
process    Thursday        ; Set this day to meet deadline ITC
                           ; gives you or Friday after
                           ; cut-over to zone based nodelist
arc 5      ; force ARC 6 or 7 to generate level 5 archives
           ; DO NOT use ARC5 with ARC versions earlier than 6

master     <directory>     ; where master files reside
                           ; (default - current)
update     <directory>     ; where to save received files
                           ; 'till processing
mailfiles  <directory>     ; where mail server places received
                           ; files
uploads    <directory>     ; where BBS places uploaded files
                           ; (if needed)
badfiles   <directory>     ; optional "waste can" for files
                           ; with fatal errors
notify  errors CRASH       ; notify of errors (CRASH optional)
                           ; Errors should not occur at this
                           ; level, but. . .
submit     1:1/0           ; where you send updates, CRASH
                           ; and/or HOLD optional
netaddress <your address>  ; your network address here
messages   <directory>     ; path name to mail server's
                           ; mail area

data

; Your zone source data goes here.  Note that first
; non-comment statement must begin with the keyword "Zone"
; not "Region"

files

Region <nbr> <generic name>
Region <nbr> <generic name>
Region <nbr> <generic name> <notification address> ; if not
                                                   ; "region/0"
Region <nbr> <generic name>
;      etc.
```

Figure 5. Sample control file for Zone Coordinators

On your process day, after generating your zone segment, you will execute **MakeNL** a second time with a special control file to

create your zone's version of the distribution nodelist. Section

```
make composite
name <network name>    ; eg. FidoNet or AlterNet or EGGnet

threshold  0 -1 ; force ARC -- no submittal difference file

arc 5      ; force ARC 6 or 7 to generate level 5 archives
           ; DO NOT use ARC5 with ARC versions earlier than 6

outfile <generic name> ; recommended names:
                       ;  NODELIST - complete composite list
                       ;  Zn-LIST  - zone distribution list
                       ;  Rnn-LIST - region distribution list
                       ;  NnnnLIST - network distribution list

outdiff <generic name> ; recommended names:
                       ;  NODEDIFF - complete composite list
                       ;  Zn-DIFF  - zone distribution list
                       ;  Rnn-DIFF - region distribution list
                       ;  NnnnDIFF - network distribution list

master     <directory> ; same as OUTPath or MASter in
                       ; previous run
outpath    .           ; put <name>.nnn in current directory

copyright <cpy-dist.txt>; if these three files are not path-
prolog    <file name>   ; qualified, they are lodated in the
epilog    <file name>   ; master directory

files

Network <nbr> <explicit name>  ; must match OUTFile name of
                               ;  previous run
;   or
Region  <nbr> <generic name>   ; must match OUTFile name
;   or
Zone    <nbr> <generic name>   ; must match OUTFile name
Zone    <nbr> <generic name>   ; file(s) received from
;  etc. . .                    ;  other ZC(s)
```

Figure 6, Sample control file for distribution lists

(?) deals with distribution lists. The only difference for ZCs is that they will need to define the MAIlfiles path to pick up the files from the other ZCs.

There are several advantages to "zone specific" distribution NODELISTs. Prologue and epilogue comments may be "zone localized." The local zone may be placed first in the list, a significant convenience to the nodes in your zone. There will probably

be quicker and better NODELIST availability.

2.6 Putting together a Distribution Nodelist Segment

Distribution lists (see section 3.7 on page 15) may be prepared at any level except Hub, though it is not recommended below Region. In theory, a distribution list may be prepared at the same time your submittal file is prepared (MakeNL will support this method). In practice, it is better to make a distribution list in a separate run of MakeNL so that the distribution list may have a distinct file name, and different copyright notices and comments.

Refer to figure 6 (a facsimile of the file DISTRIB.CTL in your distribution package). Now that you thoroughly understand all of the forgoing (you do, don't you?), this file is fairly self-explanatory. After you edit it to fit your specific needs, your MakeNL external event can use it to create your distribution files. If your MakeNL external event executes once a week, the batch file fragment is trivial:

```
MAKENL
MAKENL  DISTRIB
```

If your MakeNL external event is a daily event then:

```
MAKENL
  IF ERRORLEVEL 3 GOTO LOOP
MAKENL  DISTRIB
```

Here you make use of the fact that a return code of 3, 4 or 5 means that MakeNL was executed in test mode. That means it was not your normal process day, so the batch file simply restarts your mail server after testing any received files. On process day you will "drop through" and create distribution files.

3. Definitions

Before we proceed further, we need to define some terms in use throughout this document.

3.1 Composite Nodelist

The composite nodelist is defined, for our purposes, as the complete list of all nodes in the network. It includes all Zones, all Regions and all local nets.

3.2 Generic File Name

To MakeNL, a generic file name is one which does not have a file type (or file extension). For example, NODELIST and FILES\ZONE1 are generic file names. MakeNL does special processing on files when you specify them by generic names. It creates file types for them based on publication date and contents of the files.

The publication date is translated into a 3-digit day-of-year, represented here as "htu" where "h" is the hundreds digit, "t" is the tens digit and "u" is the units digit of day-of-year. MakeNL recognizes and/or creates the following file types for files with generic names:

.htu - Master files, uncompressed submittal and distribution files, distribution difference files
.Atu - Archived submittal and distribution files, archived distribution difference files
.Dtu - Submittal difference files
.ADu - Archived submittal difference files

3.3 Explicit File Name

An explicit file name, on the other hand, has a specified file type. PROLOG.TXT is an example. When data files are specified this way, MakeNL does only limited processing on them, specifically checking them for errors and copying them from directory to directory. It makes no attempt to deal with archives or difference files.

3.4 Archive File

An archive file is a compressed file created by the ARC program from System Enhancement Associates. You must have a copy of this program in your current directory or on your DOS path (see "PATH" command in your DOS manual) to make use of this capability of MakeNL.

The ARC program, version 6 and later, has the ability to create archives compatible with earlier versions of ARC. MakeNL provides a configuration verb to permit the use of this capability.

(See section 6.3.4 on page 30)

3.5 Difference File

A difference file is a file which defines the differences between an older version of a file and a newly created one. In fact, the difference file (sometimes called a "diff" file from the NODEDIFF naming convention) contains a simple editing language. With a copy of the old file and the difference file, MakeNL can accurately recreate the new version of file.

3.6 Submittal File

A submittal file is one which you will send forward up the ladder to be merged with other files from your level. A Hub sends his submittal file to his network host. The host sends his submittal file to his regional coordinator, etc.. MakeNL will create your submittal file for you each week, and can even create the "file attach" message to send the file over the network.

3.7 Distribution File

Distribution files, on the other hand go the other way. They are meant for distribution down the ladder. NODELIST.htu and NODEDIFF.htu are well known examples, but let's say you are a regional coordinator, and some of the nodes in your region have no interest in communicating with nodes outside the region. To conserve valuable disk space, they might wish to use a nodelist which only contains your region. MakeNL can create such files. They differ from submittal files only in the naming conventions they observe.

3.8 Absolute Directory Path

An absolute path name is one specified from the disk's root directory. It will always begin with a back-slant (\) or a drive specifier followed by a back-slant (C:\) C:\BBS and \MASTER are examples of absolute path names.

3.9 Relative Directory Path

A relative path name is one specified as a subdirectory from the current working directory, without specifying what that is. It never begins with a back-slant. FILES\RECEIVED and D:OPUS are examples of relative path names. In the latter case the path is relative to the current directory of the D drive.

See your DOS manual for additional information on path names.

3.10 Publication Date

The "publication date" is the date of the next publication of the composite nodelist. It is always the same day of the week. It is usually a Friday, but can be any day of the week.

The publication date is specifically, the next occurring publication day (today, if this is your publication day).

> NOTE: Most FidoNet compatible networks publish their nodelist on Friday, but a few publish on other days. A special verb is provided to control publication day. (See section 6.1.3 on page 22.)

4. MakeNL Command Line

MakeNL is invoked with the following command line:

```
MakeNL [<control_file_name>] [/PROCESS | /TEST]
     [/MERGE[=<list_name>] [/NAME=<composite_name>]
```

The switch names may be spelled out (/TEST), or abbreviated to as few as one letter (/T).

4.1 Control file name

By default, MakeNL expects to find a control file named MAKENL.CTL in the current directory. This can be overridden by placing the name (optionally qualified with a path) on the command line. The control file may have any file extension, but if none is given .CTL is assumed. For example:

```
MakeNL ZONE1
```

causes MakeNL to use ZONE1.CTL in the current directory, and:

```
MakeNL MASTER\NET100.CFG
```

causes MakeNL to use NET100.CFG in the subdirectory named MASTER as its control file.

MakeNL must have a control file and will abort if it can find none. See section 6 on page 21 for the contents of the control file.

In operation, MakeNL writes a number of progress messages to "standard output." By default, DOS sends these messages to your screen, but you may redirect them to a file to retain a log of MakeNL's activity:

```
MAKENL   >MAKENL.LOG

MAKENL  >>MAKENL.LOG
```

In the first case, the log file is recreated fresh each time MakeNL is executed. In the second case, MakeNL appends the results of each execution to the end of the log file.

4.2 /TEST switch

The optional "/TEST" switch, if given, will force MakeNL to operate in test mode, even on the normal process day.

4.3 /PROCESS switch

The optional "/PROCESS" switch, if given, will force MakeNL to operate in process mode, overriding the effect of a PROcess statement in the control file.

4.4 /MERGE switch

The optional "/MERGE" switch, if given, will cause MakeNL to merge your updated nodelist segment with an existing distribution nodelist to produce a temporary nodelist containing your updates.

If <list_name> is given, it must have the form:

```
/M=[path\]<generic_name>
```

See discussion of MERge control file verb on page 24 for more details.

4.5 /NAME=<network_name>

The optional /NAME switch is a specialized cosmetic switch. When you MAKe a composite list (see section 6.1.1 on page 21), the generated list begins, by default, with the line " Nodelist for. . ."

The /NAME switch may be used to insert a network name in front of the first word in that line. If the specified replacement name contains spaces, the entire parameter must be enclosed in quotes.

For example, the coordinator for a network operated by the US Parks Service might use:

```
MAKENL "/N=Parks Service"
```

The first line of his generated list would read "Parks Service Nodelist for. . ." Note that to the right of the equal sign **case is significant.** Any capitalization present will be preserved.

A FidoNet ZC might use:

```
MAKNL /N=FidoNet
```

If the MAKe statement in your control file (see page 21) is not "MAKE COMPOSITE" the /NAME switch will have no effect.

See also the NAMe verb in section (?), page (?).

4.6 Error return codes

MakeNL terminates with ERRORLEVEL (see your DOS manual) set to one of the following values:

```
  0  = Process mode - no errors encountered
  1  = Process mode - no fatal errors encountered
  2  = Process mode - one or more fatal errors encountered
  3  = Test mode - no errors encountered
  4  = Test mode - no fatal errors encountered
  5  = Test mode - one or more fatal errors encountered
254  = MakeNL aborted - I/O error
255  = MakeNL aborted - Control file error
```

5. MakeNL Directories

MakeNL uses several disk directories in the performance of its tasks. Some are required -- others are needed only to make use of special features. In this text, MakeNL's directories will be referred to by the following names.

5.1 Master Directory

The master directory is where MakeNL stores all of its master files. Received files will migrate to this directory. MakeNL requires a master directory.

5.2 Update Directory

The update directory provides interim storage for received files. It is needed only if you process files received from others, and notify them on receipt or when errors occur. It prevents multiple notification.

5.3 Mailfiles Directory

The mailfiles directory is the directory in which your mail server receives files. It is required if you receive nodelist segments from others via the network.

5.4 Upload Directory

The upload directory is the directory in which your BBS stores uploaded files. It is required if others upload nodelist segments to your BBS.

5.5 Other Directories

Depending on the options selected in the control file, MakeNL may need access to other directories.

If MakeNL will be generating network messages it must be told in which directory your mail server expects to find these messages.

By default, MakeNL places the master copy of its output file in the directory with your other master files. You can, however, define another directory to receive output files.

When a received file has a "fatal error," one which renders the entire file unusable, MakeNL must get it out of the way to prevent further attempts to process it. By default, it simply deletes the offending file, but you may define a "bad files" directory and MakeNL will move such files there.

The "current directory" when MakeNL is invoked is the default directory for MakeNL's control file.

6. MakeNL Control File

MakeNL requires a control file to define its operating parameters. By default, this file is named MAKENL.CTL, but you may specify another control file name on the command line.

The contents of the control file is a series of statements which tell MakeNL what it is to do, and how it is to do it. Each statement begins with a verb, and may have as many as three additional arguments.

MakeNL is not sensitive to "case" of characters in the control file. Upper, lower or mixed case may be used. In the discussion of the control verbs below, upper case is used to indicate the minimum abbreviation required.

The definitions of the control file verbs are grouped by type for clarity. Except for DATA and FILES, they may appear in your control file in any order.

6.1 Control statements

The following statements control the overall operation of MakeNL.

6.1.1 MAKe <segment_type> [<number> [<source_file>]]

A MAKe statement is required in every control file. It tells MakeNL what type of nodelist segment it is to produce and what kind of input to expect.

<segment_type> is one of HUB, NETwork, REGion, ZONe or COMposite.

<number> is shown above as "optional." This isn't strictly correct. For all segment types except COMposite, it is required. It must NOT be used for COMposite. <number> is the network address number associated with the segment being made.

Even if you are a Zone Coordinator, there is a part of the nodelist segment you prepare which you, yourself maintain -- the list of nodes associated directly with your segment heading. In the case of a ZONe segment type, this includes the "zone gate" nodes. In the case of an RC it is the list of "independent" nodes in your region. There are two ways to input your sub-segment into MakeNL. One is through the use of the DATa statement (see section (?) on page (?). The other is by specifying a source file in the MAKe statement.

<source_file> must be an explicit file name without a path qualifier. The named file must exist in the directory named as your master directory. The first non-comment statement

in the file must match the MAKe statement in <segment_type> and <number>.

Examples:

```
MAKE COMPOSITE      ; make a complete nodelist

MAKE NET 100 MASTER.DAT ; input data from "master"

MAKE REGION 10      ; use DATA statement
```

6.1.2 NAMe <network_name>

When you MAKe a composite list (see section 6.1.1), the generated list begins, by default, with the line " Nodelist for. . ."

With the NAMe verb, you can define a <network_name> that will be inserted in front of the first word of that line. <network_name> is the only parameter in your control file that is case-sensitive. It will appear in the list exactly the way it appears in your control file. This verb is almost identical to the /NAME switch (see section 4.5 on page 18).

There are two restrictions imposed by the NAMe verb that do not apply to the /NAME switch. <network_name> can be no more than 15 characters long, and can not contain any spaces or tabs.

If you use both the NAMe verb in the control file and the /NAME switch on the command line, the command line switch will override.

6.1.3 PUBlish <day_of_week>

Most FidoNet compatible networks publish their nodelists each Friday. For that reason, **MakeNL** assumes a Friday publication day.

As noted earlier, the file extensions **MakeNL** computes for generic file names are determined by publication date, which is defined as the next occurrence of your publication day, expressed as day of year. Obviously, your next publication date if different if you publish on Saturday, than if you publish on Friday.

The PUBlish statement let's you inform **MakeNL** that you publish your list on some day other than Friday, and he will adjust his generic file name extensions accordingly. The <day_of_week> may be a full day name, or may be abbreviated to as few as three characters.

Example:

```
PUBLISH SUNDAY      ; sets file extensions according to
                    ; Sunday's date
```

CAUTION: Some nodelist processors may fail to operate correctly with file name extensions keyed to a day of week other than Friday. For example, versions of EditNL earlier than 4.20 will never find such files. On the other hand, XLATLIST handles all such extensions, but has a problem with the first list of a new year if you don't delete the list(s) from the previous year first.

6.1.4 PROcess <day_of_week>

MakeNL has two fundamental modes of operation, "test" and "process."

In test mode, it scans input data and checks it for errors and it moves received update files to the update directory. Unless the MERge statement or the /M command line switch is used, it does not produce any output.

In process mode MakeNL produces an updated master file for your nodelist segment and any required submittal or distribution files. Normally you will do this once each week.

If no PROcess statement is present, MakeNL will operate in process mode unless the /T command line switch is used. If the PROcess statement is present, it controls MakeNL's mode of operation. Except on the day specified in this statement, MakeNL will operate in test mode. When MakeNL is executed on the defined process day, it operates in process mode.

This statement is most useful to coordinators who receive update files from others. It permits them to operate MakeNL daily in an "external event" of their mail server. It error checks received files and sends back notification on a daily basis as files are received, but only on the appropriate day does it create a submittal file and forward it up the ladder.

The <day_of_week> may be a full day name, or may be abbreviated to as few as three characters. It should be chosen so that you may meet the submittal deadline set by the next higher coordinator.

Example:

```
PROCESS THURSDAY
```

6.1.5 MERge [<nodelist_name>]

When you add new nodes (or nets, as the case may be) to your list, you might like to notify them immediately that their applications have been acted upon. This is complicated by the fact that your mail server needs to know about these new nodes (or nets), and that information comes from the distribution nodelist. It may be several days before you receive the updated version of the distribution nodelist containing the new listing.

The MERge statement tells **MakeNL** to merge your updated segment into an existing distribution nodelist to create a temporary nodelist your mail server can use. This action will occur even if **MakeNL** is operating in test mode.

By default, **MakeNL** will search for NODELIST.htu not more than two weeks old in the current directory, create a temporary list named NODELIST.999, copy from the distribution list to the temporary list until it finds your segment in the distribution list, generate an updated version of your segment in the temporary list and finally, copy the remainder of the distribution list to the temporary list.

In order to find your segment in the distribution list, **MakeNL** uses information from the MAKe and NETaddress statements in your control file.

If you are MAKing a HUB, NETaddress must provide the net number. If no NETaddress is given, the MERge command will be canceled.

If you are MAKing anything other than a ZONE, and if the distribution list is a zone-based list, the zone number from the NETaddress command is used to place your segment in the correct zone. If no zone number is given, **MakeNL** assumes that your segment goes in the first zone it encounters in the distribution list.

If present, <nodelist_name> must be a generic file name, but may be qualified by a path name. It overrides the name and location of the distribution list, but the rest of the actions are the same. The temporary list is placed in the same directory as the distribution list being merged.

If the MERge statement is used in your control file, it will ALWAYS be executed. If you prefer to "merge on demand" use the /M command line switch instead. It is exactly equivalent to the MERge statement. (See section 4.4 on page 18.)

Example:

```
MERGE \BBS\NODELIST
```

6.1.6 PRIvate <disposition>

Private nodes (identified by the PVT keyword in the nodelist) are not universally permitted by all networks. In a commercial network, for example, a node with no phone number listing is not particularly useful. By default, MakeNL will accept these node entries contrary to a network coordinator's intentions.

The PRIvate statement permits you to define other dispositions for such entries. <disposition> must be one of OK, NONE, ZONE, REGION, HOST or HUB.

"PRIvate OK" is equivalent to the default action -- private nodes are permitted (but, of course, only within nets).

NONE says they are not permitted. The PVT keyword will be treated as an error, with a note that "Private nodes are not permitted in this network."

If <disposition> is any of the hierarchical keywords, a private node will be transformed into an "open" node by removing the PVT keyword and substituting a known phone number in the phone number field. The number used is that of the nearest node at least as high in the hierarchy as the keyword specified. For instance, if "PRIvate HOST" is given, a private node will become a normal node with the phone number of the most recently encountered Host, Region or Zone statement. Even if the previous entry were a Hub, his phone number would not be used unless "PRIvate HUB" had been given.

For fairly obvious reasons, if you are MAKing a NETwork and enter the statement "PRIvate ZONE," MakeNL will abort with a control file error. It must be able to find at least one phone number to make the substitution, and your file can't have any zones or regions in it! The hierarchical level of <disposition> must be equal to or lower than the level you are MAKing. In the case you MAKe COMPosite, MakeNL will take your word for anything your PRIvate statement tells it, but if no matching phone number has been found before it encounters a private node, the private node will be treated as though the statement were "PRIvate NONE," and will generate an error.

6.1.7 MINphone <minimum_parts>

The sixth field of a nodelist entry is the phone number field. MakeNL insures that it meets certain minimum standards. It must begin and end with a digit. It must contain only digits and hyphens. There must not be two hyphens in a row.

Phone numbers usually consist of several parts, separated by

hyphens. For example, a number might have a country code,

and area code, an exchange code and a line number. To MakeNL, a phone number is a phone number. It attaches no special meaning the number's various parts.

Some nodelist processing programs do attach meaning to the parts of a phone number, and complain if they are not there! For this reason MakeNL will enforce an additional constraint, if told to do so. It will insure that every phone number has a certain minimum number of parts.

Example:

```
MINPHONE 4
```

This statement in a control file instructs MakeNL to reject any entry with a phone number that does not have at least four parts. The number "123-4567" would not be valid and would cause an error message to be issued.

In a large hierarchical network of international scope, the MINphone verb should only be used at the lower levels. Hubs and Hosts may expect all phone numbers to fit some standard format. Except in North America, regional coordinators are likely to find variability in phone number format from country to country. A zone coordinator will certainly encounter such variability. Attempts to enforce a standard number of parts in a phone number at these levels will not be very successful.

This is not a serious constraint on the use of MINphone. The files received by RCs and ZCs have already been processed at least once by MakeNL at a lower level where format can be enforced. There should be no need to do it again.

6.1.8 BAUdrate <valid_baud_rates>

The seventh field of a nodelist entry is the baud rate field. MakeNL checks the value in this field against a table of valid baud rates. Experience has shown this to be necessary. Certain types of errors can cause a part of the phone number to appear to be in the baud rate field. Without a validity check, these errors could go undetected. By default, the valid rates are 300, 1200, 2400, 4800, and 9600.

To accommodate future growth, the "valid baud rate" table is configurable through the use of the BAUdrate verb. The <valid_baud_rates> parameter is a list of up to twelve (12) valid baud rates (numeric strings) separated by commas. It may contain no spaces of tabs. It must be the complete list, and will replace MakeNL's internal table.

For example:

BAUDRATE 1200,2400,4800,9600

This statement would effectively eliminate 300 baud nodes from your list, while:

BAUDRATE 300,1200,2400,4800,9600,14400

would permit nodes to use 14400 baud in their entries. A caveat in this case. Any new values must be authorized from the top down. If you used this statement and submitted a list with 14400 baud nodes, the first coordinator who did not have a similar BAUdrate statement would reject those nodes as having invalid baud rates.

In general, you can be more restrictive than the coordinator(s) above you, but you **cannot be more liberal!**

6.2 Defining nodelist segment directories

MakeNL uses several directories for various purposes for storage of nodelist segment data. The following statements are used to define to **MakeNL** where these directories are located. They all have the same format:

<verb> <directory_path_name>

<verb> is one of MASter, UPDate, MAIlfiles, UPLoads or BADfiles as defined below.

<directory_path_name> is a relative or absolute directory path name, with or without a disk specifier.

6.2.1 MASter <directory_path_name>

This is the directory in which **MakeNL** keeps all its "master" files. This includes the master copy of your submittal file, master copies of nodelist segments received from others, your input source file, if used, and the three "comments" text files, if used. In addition, for each segment given a generic name, week-old copies are kept here and used create and/or apply difference files.

MakeNL always uses a master directory. If this statement is absent, the current directory is used as the master directory.

6.2.2 UPLoads <directory_path_name>

If others upload update files to your BBS, this statement tells **MakeNL** where to look for them. Only one upload directory may be named. If your BBS supports more than one, you

need to inform your submitters how to upload their updates so that they wind up in the right directory.

If you don't receive update files via upload, omit this statement and **MakeNL** will ignore it.

6.2.3 MAIlfiles <directory_path_name>

If you receive update files from others via network mail, this statement tells **MakeNL** where your file server will put the files.

If you don't receive update files via network mail, omit this statement and **MakeNL** will ignore it.

6.2.4 UPDate <directory_path_name>

If you receive update files from others by any means, this statement defines a temporary holding directory for them. When **MakeNL** is operated in test mode, files it finds in the UPLoads or MAIlfiles directories are moved to this directory. If you use notification (and you should use at least error notification), **MakeNL** will generate a notification message each time it finds a submitter's update file in one of those directories. Moving received files to the update directory prevents multiple notification.

If received files were moved to the master directory in test mode, and if a subsequent difference file were received from one of your submitters, you might have already lost the file **MakeNL** needs to apply the difference file against!

When **MakeNL** operates in process mode, it moves files found in any of these directories to the master directory as part of its update process.

If you don't receive update files from others, omit this statement and **MakeNL** will ignore it.

6.2.5 BADfiles <directory_path_name>

When a received file generates a fatal error, **MakeNL** refuses to process it further. By default, it simply deletes the offending file, but if this directory is defined, **MakeNL** will move files with fatal errors here. This gives you a chance (if you feel so inclined) to edit the file to correct the error, then move it back to say the MAIlfiles directory and process it again.

If you don't receive update files from others or don't want to correct their errors for them, omit this statement and **MakeNL** will ignore it.

6.3 Defining output

The following statements tell **MakeNL** what output to generate and how to generate it. They only have effect when **MakeNL** operates in process mode.

6.3.1 OUTFile <filename>

This statement is required in your control file. It defines the name of your master and submittal files. These are the same file unless **MakeNL** creates an archive or a difference file. These files are placed in your OUTFile directory, or if it is not defined, your MASter directory.

<filename> may be either explicit or generic, but the choice carries some implications with it. Using an explicit output file name inhibits some of **MakeNL**'s features, but if your output file is small they may not be of interest to you. Specifically, **MakeNL** will create neither an archive file nor a difference file for explicitly named files. The reason is that to do so, **MakeNL** must have the freedom to manipulate file extensions, but it cannot do that if you give it an explicit file extension.

6.3.2 OUTPath <directory_path_name>

By default, **MakeNL** keeps all of its output (except a merge file) in the MASter directory. This includes the output master file, the submittal file (if different), and distribution files. The OUTPath statement, if present, tells **MakeNL** to keep its output in a different directory.

6.3.3 THReshold <arc_size> [<diff_size>]

When you use a generic output file name, **MakeNL** will do its best to insure that your submittal file is small enough to send to your coordinator in a one-minute phone call. It has two ways to compress a file.

First, it can create an archive file, using the ARC program. This will typically achieve about 40% reduction in size.

Second, it can create a difference file. This is a file which contains only the differences between your new output file and your previous one. This technique can achieve dramatic file size reduction, but is not as safe as the former. It assumes that your coordinator has a copy of your previous file (usually a safe assumption). With it and the new difference file he can reconstruct your new output file.

At 2400 baud, you can transmit a file about 10000 characters long in a one-minute phone call. If an archive file is at least 40% smaller than its uncompressed version, a file of

up to 16666 characters can be archived and transmitted in a

minute. These are the two default thresholds MakeNL uses to determine what to do with your submittal file. The THReshold statement can override these defaults.

If your submittal file is larger than <arc_size> (in bytes), MakeNL will use ARC to create an archive of it, and submit the archive file.

If your submittal file is greater than <diff_size>, MakeNL will create a difference file and submit it. If a difference file is still larger than <arc_size>, MakeNL will create an archive of the difference file and submit that, but that is the best it can do.

If <diff_size> is given, it must be greater than or equal to <arc_size>. If <diff_size> is not given it is calculated by MakeNL as 5/3 * <arc_size>.

Examples:

```
THRESHOLD 5000 ; recommended value for 1200 baud

THR 0          ; always create an archived diff file

THR 0 -1       ; always archive -- no diff

THR -1 -1      ; turn compression off
```

6.3.4 ARC <n>

MakeNL creates compressed archive files by executing the program ARC by System Enhancement Associates.

ARC will, by default, produce the maximum compression of which it is capable. This may not always be desirable. If you have version 6 or 7 of ARC, and the recipient of your archived file does not, he will be unable to unpack your archives.

The ARC <n> verb will force these versions to create level-5 or level-6 compatible archives, thus insuring compatibility. The parameter <n> must be either "5" or "6."

Do not use ARC <n> if you have ARC version 5 or earlier. It will cause errors, as will the use of "ARC 6" with version 6! The compatibility level must always be LOWER that your version level.

6.3.5 OUTDiff <generic_filename>

MakeNL's primary mission is to gather segments of the nodelist and assemble larger segments for submittal to the next higher coordinator, until finally all the segments have

been assembled and the result is the network's distribution

nodelist. However, if desired, distribution lists may be created at any step along the way.

Distribution lists are actually created in pairs. A familiar pair is NODELIST.htu and NODEDIFF.htu, the files which define the Public FidoNet Network.

There is no reason why a node must use the full network list if a smaller list will suit his needs. Indeed, programs such as PRUNE are based on this philosophy.

When the OUTDiff statement appears in your control file, **MakeNL** will produce a pair of distribution files. OUTFile and OUTDiff must specify distinct generic names, and they will become the "stem" names of your distribution files.

Distribution files differ from submittal files in the naming conventions used to determine file extensions. For example, assume:

```
OUTFILE REGION10
OUTDIFF R10-DIFF
```

Then the following table defines the file names of generated submittal and distribution files:

Compression	Submittal	Distribution
None	REGION10.htu	REGION10.htu
Archive	REGION10.Atu	REGION10.Atu
Difference	REGION10.Dtu	R10-DIFF.htu
Both	REGION10.ADu	R10-DIFF.Atu

Note that the full list and its archive bear the same name in both columns. Indeed, they are the same files. But submittal files use the same filename stem for all types of files, while distribution difference files have unique filename stems. The reason for this is to support the traditional names NODELIST and NODEDIFF as painlessly as possible.

Distribution files, as with submittal files, are placed in the OUTPath directory. Once created, these files are ignored by **MakeNL**, unless explicitly told to clean up its old files (see the CLEanup command below).

The following control file fragment would create NODELIST distribution files for the Public FidoNet Network.

Example:

```
MAKE COMPOSITE
THRESHOLD 0 -1 ; force archiving -- no submittal diff
OUTPATH  .        ; master list to current directory
OUTFILE  NODELIST ; create NODELIST.Atu
OUTDIFF  NODEDIFF ; create NODEDIFF.Atu
```

When MakeNL has finished, three files are left in the current directory; NODELIST.htu, NODELIST.Atu and NODEDIFF.Atu. Note that with "THRESHOLD 0 0," NODELIST.ADu would be created with contents identical to NODEDIFF.Atu.

6.3.6 CLEanup

If you use a generic OUTFILE name, and if you create difference and/or archive files, they tend to accumulate in your OUTPATH directory over time. By default, MakeNL ignores them completely. This is done so that OUTPATH can point to a "download" directory where several editions might be made available to users.

When MakeNL operates in process mode and CLEanup (no arguments) appears in the control file, after all other processing is done, it searches the OUTPATH directory for old difference and/or archive files and deletes any it finds.

CLEanup has no effect on current files or the handling of output master files. It has no effect whatever when MakeNL is operated in test mode.

6.4 External message transmission

MakeNL can generate two kinds of network messages, notification messages and submittal messages.

Notification messages are addressed to submitters to notify them of receipt of and/or errors in their submittal files.

A submittal message is addressed to your coordinator, and has your submittal file attached.

Messages created by MakeNL are compatible with SEAdog by System Enhancement Associates. Except for CRASH and HOLD attribute flags, they are compatible with Fido v11, and (I think) Fido v12 by Tom Jennings. They are probably compatible with other mail servers, but I cannot guarantee it.

The following statements control the generation of these messages.

6.4.1 NETaddress [<zone>:]<net>/<node>

This statement is used to tell **MakeNL** what your network address is. **MakeNL** must know your network address in order to generate the "From:" field in submittal or notification messages. If **MakeNL** must create inter-zone messages, it needs to know your zone number. In addition, your zone number, and if you are MAKing a HUB, your net number may be needed to properly perform a MERge operation.

If your control file does not contain a NETaddress statement, **MakeNL** will try to guess you network address from your MAKe statement. If you are MAKing a ZONE, your default network address is <nbr>:<nbr>/0. If you are MAKing a REGION or NETWORK, your default network address is 0:<nbr>/0. If you are MAKing a HUB, **MakeNL** hasn't a clue!

If the default network address **MakeNL** selects for you is satisfactory, you don't need a NETaddress statement, but it is recommended that you include one to remove all doubt.

Examples:

```
NETADDRESS  100/76

NETADDRESS  1:1/0
```

6.4.2 MESsages <directory_path_name>

This statement is required if **MakeNL** is to generate ANY outgoing messages. It tells **MakeNL** the name of the directory used by your mail server for network mail messages.

Example:

```
MESSAGES  D:\BBS\MAIL
```

6.4.3 SUBmit <address> [<flags>]

The SUBmit statement tells **MakeNL** that in process mode it is to create a "file attach" message to send your submittal file to your coordinator. <address> is his network address.

Optional <flags> may be any or all of CRASH, HOLD or INTL. These should not be used unless your mail server supports them.

Example:

```
SUBMIT  13/0  CRASH
```

The INTL flag is a special case. It forces **MakeNL** to add the "^AINTL: <from> <to>" extended address line to <u>all</u>

messages it creates, even if they are not leaving your zone.

Indiscriminate use of this flag is not recommended. You should use it only if your mail server needs all messages to contain the expanded address.

NOTE: If you don't know what I'm talking about, don't use the INTL flag. It may cause you trouble. Consult your mail server's documentation for guidance. Most mail servers do not require this feature and many consider these messages malformed. Two which may need the INTL flag are D'Bridge and FrontDoor.

6.4.4 NOTify <type> [<flags>]

If the NOTify statement is used, when MakeNL finds a received update file in either the UPLoads or MAIlfiles directory, it will notify the submitter that the file was received and/or that the file contained errors.

Optional <flags> may be any or all of CRASH, HOLD or INTL. These should not be used unless your mail server supports them. (INTL need only be specified once, and it applies to all messages.)

<type> is one of RECeipt, ERRors or SELF.

RECeipt and ERRors control notification of nodes submitting files to you. If only NOTIFY RECEIPT is used, all notification messages will have the same attributes, as defined by <flags>. If only NOTIFY ERRORS is used, the submitter is NOT notified unless his update file has errors. NOTIFY RECEIPT and NOTIFY ERRORS may both be used with different message attribute flags, if desired. In other words, normal receipt notification may be sent "regular" mail, while error messages are sent "CRASH."

Unless you execute MakeNL with the "/T" switch, NOTIFY SELF instructs MakeNL to enter a message to you if it encounters any errors in your input data. If you edit your master file without testing it and you goof (shame), the error could go unnoticed for some time. When MakeNL is executed by your external event, the error will cause a message to you to appear in you network mail area.

6.5 Defining special files

MakeNL uses a number of special files in the performance of its duties. The statements in this section provide the means to define these files and/or alter defaults.

6.5.1 Output comments files

When MakeNL processes nodelist segment data, it ignores comments embedded in the data files. These comments are as-

sumed to be your notes to yourself. They might be anything

-- voice phone numbers, suspense dates when "down" nodes are expected to come back on line, etc..

Comments may be placed in submittal files to communicate information to your coordinator, or in distribution files to communicate information to your users, but **not** by embedding them in your data.

Instead, comments are placed in your output files by creating and maintaining three special text files in your MASter directory. In process mode, **MakeNL** searches for these files and, if they exist, copies them into your submittal file. It is **not** an error if any of them does not exist. Missing comments are simply ignored.

When **MakeNL** copies any of these files to the output file, it examines the first character of each line to see if it contains a "comments flag," a semicolon (;) character. Any line beginning with a semicolon is copied as is to the output file. **MakeNL** inserts ";S " at the beginning of any line which does not begin with a semicolon.

6.5.1.1 COPyright <explicit_filename>

By default, **MakeNL** looks for a file named CPYRIGHT.TXT in the MASter directory. If it exists, it is assumed to be a copyright notice, and is copied into the output file immediately under the identification header line.

The copyright file gets a special treatment the other comments files do not get. If a string of four pound signs (####) is found in the first line, it is replaced with the current year. Thus "Copyright ####" becomes "Copyright 1987" automatically. You don't have to remember to edit your copyright notice in January.

The COPyright statement lets you override the default name "CPYRIGHT.TXT" with any other explicit name. You might need to do this if, for example, you are a Regional Coordinator and use two different control files to create your submittal file and distribution files. You may use the same MASter directory for both and have different copyright notices on each. In fact, you might name a nonexistent file when generating your submittal file, thus eliminating the notice.

Examples:

```
COPYRIGHT  CPY-DIST.TXT

COPYRIGHT  NONE  ; file "NONE" does not exist
```

6.5.1.2 PROlog <explicit_filename>

By default, **MakeNL** looks for a file named PROLOG.TXT in the MASter directory. If it exists, it is copied into the output file immediately under any copyright notice.

The PROlog statement allows you to override the default filename, "PROLOG.TXT" with any explicit file name. If the named file does not exist in the MASter directory, the effect is to turn off prologue comments.

6.5.1.3 EPIlog <explicit_filename>

By default, **MakeNL** looks for a file named EPILOG.TXT in the MASter directory. If it exists, it is copied into the output file following all nodelist segment data.

The EPIlog statement allows you to override the default filename, "EPILOG.TXT" with any explicit file name. If the named file does not exist in the MASter directory, the effect is to turn off epilogue comments.

6.5.2 Comments from received files

As stated earlier, comments in nodelist segment data files are NOT copied into output files. But comments in your received files are probably there to communicate information to YOU. Like the old dairy cream separator, **MakeNL** can separate comments from received files out into their own file.

6.5.2.1 COMments <explicit_filename>

When the COMments statement appears in your control file, it tells **MakeNL** to write the name of each input file to the named comments file, followed by any comments statements that appear in that file.

The file name MUST be explicit, but may have a path name. The comments file will appear in the current directory if no path is given.

If the COMments statement is omitted, comments in received files are ignored by **MakeNL**.

6.6 DATa

The DATa statement is used as an alternative method of entering your nodelist segment (see MAKe on page 21). If used, the DATa statement MUST follow all control statements except FILes.

The DATa statement actually causes **MakeNL** to shift gears. It has been processing control information. After the DATa statement, it begins processing nodelist data, which immediately follows it

in your control file. The format and rules for this data are the same as for any nodelist data.

The first non-comment statement encountered must match the MAKe statement in <segment_type> and <number>, or a fatal error will occur.

6.7 FILes

If you receive segment updates from others, you must use the FILes statement to define the files you receive and process. If you furnish all the input data yourself, you will not use a FILes statement.

When used, the FILes statement MUST be the last control statement in the control file. It is followed by a list of file entry statements in the following format:

```
<segment_type> <number> <filename> [<notify>]
```

<segment_type> is one of HUB, NETwork, REGion or ZONe, but it must be hierarchically lower than the MAKe <segment_type>. In other words, if MAKing a region segment, <segment_type> must be either NETwork or HUB -- it may not be REGion or higher.

<number> is, of course the network address number associated with the segment.

<filename> is used by **MakeNL** to search through the various directories for the file defining this segment. You and your submitter must agree on this filename. His **MakeNL** must generate what your **MakeNL** will look for. <filename> may be either explicit or generic, but it must be defined the same by both you and your submitter.

The first non-comment statement in an input segment file must match <segment_type> and <number>, or a fatal error occurs.

When you use notification, **MakeNL** will calculate a network address for notification messages. If you are MAKing a network segment for, say net 107, and receiving update segments from your Hubs, notification messages will be sent to 107/<number>. In all other cases, **MakeNL** sends notification messages to <number>/0 by default.

The optional argument <notify> is the network address **MakeNL** should use for notification messages and will override the calculated address.

Examples:

```
REGION    14   REGION14       14/61
NETWORK  100   NETWORK.100
```

7. Disk and Memory Space Considerations

If you do not use archiving compression, MakeNL should execute satisfactorily in about 160 kilobytes. If you are using archiving, you should have about 290 kilobytes of memory. IF YOU ATTEMPT TO EXECUTE ARC WITH TOO LITTLE MEMORY, YOU MAY LOSE FILES!

MakeNL is not as extravagant with disk space as it sounds, but if you are a Regional or Zone Coordinator, you should do a little planning.

Your MASter directory will hold one copy of each .TXT comments file you use (plus backups if your editor creates them), one copy of each data file with an explicit name, two copies of each data file with a generic name, and, if you use it, one copy of your master input file. MakeNL automatically manages the data files in the MASter directory so that this is true. Obviously, there can only be one copy of a file with an explicit name in any directory. Generically named data files more that two weeks old are automatically deleted. (The most current file is always kept, no matter how old it is.)

Your OUTPath directory (usually the same as your MASter) will hold one copy of your output master file if it is explicitly named, and two copies if it is generically named. In addition, when MakeNL executes in process mode, it creates a temporary new output master file, and the OUTPath directory must have room to hold it. At least one of these copies (and occasionally two, if there were no changes) will be deleted before MakeNL attempts to do anything with or about your submittal file.

If your submittal file is to be compressed, the compressed copy will also be placed in the OUTPath directory. On completion, there will be only one submittal file, but if, for example, an archived diff file is being produced, there must be space enough for both until ARC completes. Then the diff file will be deleted.

It is conceivable (though most unlikely) that you could have a copy of each of your data files in each of the subdirectories UPLoads, MAIlfiles and UPDate. It is probable that by the end of each process cycle you will have at least one copy of at least half of these files in one of these directories. When a file is received in compressed form, MakeNL decompresses it in the directory that received it, and the compressed version is deleted. When files are moved from one directory to another on the same drive, they are moved by renaming them. No additional space is required for this operation.

"So how much !@#$%^ space do I need!?" If you are a Hub or network coordinator, not much. Here is a guide I would suggest for

Regional or Zone coordinators. Make sure you have three times

the space required for one set of your generically named files plus two times the space required for one set of your explicitly named files.

With experience, you'll get a better feel for how much space you need to operate **MakeNL**. Before cutting yourself too fine, remember that **MakeNL** needs some temporary working space while executing.

8. Contents of MakeNL Distribution Package

The **MakeNL** distribution package consists of the following files:

MAKENL.EXE	The **MakeNL** program
MAKENL.PRN	This documentation file
HUB.CTL	Sample control file for Hub Coordinators
NET-S.CTL	Sample control file for Network Coordinators
NET-L.CTL	Sample control file for Network Coordinators
REGION.CTL	Sample control file for Regional Coordinators
ZONE.CTL	Sample control file for Zone Coordinators
DISTRIB.CTL	Sample control file for Distribution Lists
CPYRIGHT.TXT	Sample copyright notice
PROLOG.TXT	Sample prologue file
EPILOG.TXT	Sample epilogue file
FTS-0005.TXT	Nodelist Format Specifications

9. MakeNL License Information

MakeNL is distributed under the "shareware" concept. It may be freely copied and distributed provided all the files listed in section 8 are included in the distribution. But **MakeNL** is **NOT FREE.**

If you are a non-commercial user of **MakeNL,** you are asked to support its development and maintenance both with your suggestions and bug reports and with your dollars. No particular fee is set for non-commercial use. Let your conscience be your guide.

Commercial users, and users participating in a commercial network are required to pay a license fee of twenty five (25) U. S. dollars for each computer on which **MakeNL** is installed for use.

The Coordinator of a commercial network may obtain a network license for a one-time fee of $1,500 US. This will license all current and future nodes of the network for the use of **MakeNL.** **MakeNL** and **EditNL** may be licensed as a set for a one-time fee of $2,250.

Please address all correspondence to:

Ben Baker
Baker & Associates
One Mark Twain Plaza, Ste 325G
Edwardsville, IL 62025

(618) 656-8850