```
____  ___  ___ ____  __
 /   /  / /__/  /   /   \/
/   /__/ /   __/__ /__  /\
```

The Maximus-Compatible Topic Extraction Program

TopicX has been designed to provide a high-quality topic extraction service for SysOps of systems with a Fido or Max -compatible message structure. TopicX is the exclusive property of Scott Dudley, and in no way shall it or the accompanying documentation be distributed in any modified form. Full credit should be left to Scott Dudley, as the developer and creator of this utility.

Table of Contents

WHAT TOPICX DOES

TopicX is a utility program written for systems with a Fido or Max -compatible message structure. TopicX allows extraction of messages based on keywords found within each message. TopicX extracts the messages to a flat ASCII file (and optionally inserted into an archive), which can be read or downloaded by users. TopicX is much more than a simple find utility -- TopicX has a set of sophisticated options which allow you to do everything from the normal to the sublime. Since time is valuable, especially when searching large EchoMail areas, TopicX has been hand-optimized to give the fastest performance possible, while searching for and extracting messages. TopicX is perfect for systems that carry a large number of EchoMail areas: The SysOp can create a configuration file that will create concise topical extracts from each EchoMail area, and make them available to users for downloading. In this way, a large volume of extracts can be collected, and made available to the general public.

Not only can a user find the information s/he wants on a particular topic FAST, but they don't have to wade through all of the "noise" usually involved with the average EchoMail conference. The extracts can also be compressed using your favorite archiving program and inserted into a download area, which reduces users' on-line time even further. Not only does TopicX make things easier for your users, the configuration file is also easy for the SysOp to maintain. By keeping all the information in one centralized control file -- rather than three or four -- TopicX allows the SysOp to maintain a topic extraction system with a minimum of effort. Although TopicX is primarily EchoMail oriented, there are several features that allow TopicX to be used in other situations as well. EchoMail is a rich flow of information, that TopicX can compile to create interesting and informative extracts.

SYSTEM REQUIREMENTS

TopicX requires an MS-DOS or PC-DOS based system running MS/PC-DOS version 3.0 or higher. TopicX also requires a maximum of 128K free RAM, plus (optionally) enough memory to load and run your favorite archiving program. To use TopicX effectively, your system must use a Fido or Max -compatible message system. It may be possible to convert other message base types, but that is beyond the scope of this document.

STARTING TOPICX

TopicX is capable of being run from the DOS prompt, or from within a batch file. The extraction of messages is configuration-file based, although there are some options which must be specified on the command line. The format for starting TopicX is:

```
TOPICX [[-X]...]
```

where -X is an optional command-line parameter.

COMMAND-LINE PARAMETERS

TopicX accepts seven different command-line parameters, which are all case-insensitive. Parameters supported by TopicX are:

-C Specifies the name and location of the configuration file. If no configuration file is specified, then TOPICX.CFG in the current directory is used by default. If the configuration file does not exist, or if an invalid command-line switch was specified, TopicX will abort with an errorlevel of 1. Example: TOPICX -Cc:\max\topicx.cfg

-A Specifies that TopicX should scan all messages in each area. This is the default, and will be used if neither -T nor -Y is specified.

-T Specifies that TopicX should only scan messages that were entered/received today. This option is not recommended, for reasons detailed below.

-Y Specifies that TopicX should only scan messages that were entered/received yesterday. This is the best option to use if you run TopicX once per day.

-F Specifies fast scanning of messages. This switch should only be used in conjunction with either -T or -Y, and TopicX will warn you if it isn't. This option tells TopicX to look at the DOS datestamp of each message, instead of the internal datestamp inside each message. This is provided as a command-line option, since it may not work correctly with some message-base renumberers or reply-chain linkers. However, it has been tested and is known to work with Renum 3.3, Renum 4.1, ConfMail Renum, ConfMail Maint, and ReplyLnk.

-P Specifies that you wish to compile the configuration file into a format more quickly usable by TopicX. See the section entitled "Pre-Compiled Configuration Files" for more information on how this option works.

-D Turns the "Date Debugging Mode" on. This option triggers the display of dates contained in each message processed. This option isn't normally needed, and should only be used when running TopicX manually. See the "Troubleshooting" section for more details.

If you are running TopicX once per day, it is advisable that you either enable the low/high water marks (covered in a separate section), or use the -Y parameter. Using this ensures that no message will be extracted twice,

and that all messages containing the specified keywords will be extracted. If you use the -T parameter instead, there is a chance you will miss messages entered/imported later in the day after TopicX is run, so that switch isn't recommended. There are also options provided to run TopicX more frequently than once a day -- See the section on Low/High Water Marks.

CONFIGURATION FILE FORMAT

TopicX uses an ASCII-based control file to determine where and how to extract messages. Each line in the configuration file must be terminated by a carriage-return/linefeed pair, and each line must be no longer than 255 characters. TopicX will try to detect either of the above conditions, and issue an error message.

The ';' character is used as a comment character, and may be inserted anywhere in the configuration file. Any text on the same line following a comment character will be ignored. Most options in the configuration file are specified in the following format:

```
<verb> [[<parameters>]...]
```

Parsing a large configuration file may take a little bit of time, so please be patient. Also note that almost all options in the configuration file are case-insensitive; Those which are case-sensitive are clearly labelled.

PRE-COMPILED CONFIGURATION FILES

If you specify the -P command-line parameter, it is possible to compile a configuration file to an intermediate form, which is more quickly usable by TopicX. It may reduce the parsing time of a large control file to one tenth of the time required to parse an uncompiled version. The compiled configuration file will use the same "stem" as the normal configuration file you specify, but will use a .DAT extension. TopicX will regenerate the .DAT file if it detects the date on the ASCII control file has changed, or doesn't exist. Otherwise, TopicX will read the pre-compiled configuration information instead. The -P parameter must be used both to generate and use the pre-compiled configuration file.

INCLUDING DATE INFORMATION IN CONFIGURATION FILES

For creating "timely" extracts, there is a method that allows you to insert controls characters into the configuration file, which translate into the current date and time. TopicX will substitute the date on which TopicX started execution throughout the entire file: In other words, the minutes and seconds will be the same for all occurrences of each parameter in the configuration file. However, these control characters are only allowed in the following locations:

o Normal keywords. Date information may NOT be included in regular expressions (topics which specify the 'R' flag).
o Output, archive and message area paths.

The format of the date/time codes is a percent sign, followed by a CASE-SENSITIVE identifier. The identifiers, and their translations, are as follows:

%a - The abbreviated weekday name
%A - The full weekday name
%b - The abbreviated month name
%B - The full month name
%c - Standard date and time string (mm-dd-yy hh:mm:ss)
%d - Day-of-month as a decimal (01-31)
%H - Hour, in the range of 00-23
%I - Hour, in the range of 01-12
%j - Day-of-year as a decimal (001-365)
%m - Month as a decimal (01-12)
%M - Minute as a decimal (00-59)
%p - Local AM or PM as a string
%S - Second as a decimal (00-59)
%U - Week-of-year, Sunday being the first day (00-52)
%w - Weekday as a decimal (0-6), Sunday being 0
%W - Week-of-year, Monday being the first day (00-52)
%x - Standard date string (mm-dd-yy)
%X - Standard time string (hh:mm:ss)
%y - Year as a decimal w/o century (00-99)
%Y - Year, including century as a decimal
%Z - Time zone (Must have TZ environment variable defined)

To use an actual percent sign in the text of the configuration file, either use two percent signs in a row ("%%"), or see the following section regarding quoting.

QUOTING

Since TopicX uses several commonly-used ASCII characters as configuration-file delimiters and comment characters, it becomes obvious that there must be a way to specify such a character as the topic of a search. If you wish to specify a "special" character, such as mentioned above, simply prefix it with a backslash. For example, instead of searching for "Hello; Look over there!", you would search for: "Hello\; Look over there!". Similarly, to use an

actual backslash, use two backslashes in a row instead. These quote characters are only used inside keywords -- Outside of keywords (such as in an output path), the quote character will be taken for a literal backslash. The only instance in which the quote character will NOT work is in conjunction with the macro character. To use a literal '@', use two "at" signs in a row (ie. "@@").

CONFIGURATION FILE VERBS

What follows are descriptions and implementation notes for all of the currently available configuration-file verbs. If you enter an invalid verb or incorrect parameter, TopicX will exit with an errorlevel of 2.

Verb:
    LOG

This option tells TopicX where to log its run-time output to. The log details such occurrences as errors, message areas processed, messages processed, etc. The output is in an Max -compatible format, and may be included in a normal system log (MAX.LOG) if desired. If no log file is specified, then none will be used, and TopicX will only output to the local console.

Example:

```
    Log C:\Max\Max.Log
```

Verb:
    ARCHIVER

Using this verb, TopicX allows you to take the ASCII extracts it generates, and insert them into a compressed archive file. This verb also tells TopicX which archiving program to use, and what command-line arguments to use when executing it. Almost any archiving program can be used in conjunction with TopicX, as long as you specify the command-line arguments correctly. This function is optional, your message extracts can remain in an uncompressed form if you so desire.

TopicX will pass the string following the Archiver keyword directly to DOS, with the exception of two run-time translations: If TopicX finds the token '%archive%' in the text to be passed to the archiver, it will replace it with the name of the archive to be added to. If TopicX finds the token '%extract%' in the text to be passed to the archiver, it will be replaced with the name of the extract to insert into the archive.

Examples:

```
    Archiver LHarc m %archive% %extract%
       or
    Archiver PKzip -aex %archive% %extract%
       or
    Archiver PKarc u %archive% %extract%
```

Verb:
    ORIGIN

The Origin verb controls handling of origin and tearlines in messages. Using the command 'Origin NoPrint',

TopicX will disable the exporting of origins/tearlines to the ASCII extracts. Using the command 'Origin NoScan', TopicX will not include the origin/tearline as part of its message body searches, which means it won't serach for keywords past the origin line. Using 'Origin None' will neither print nor scan the origin/tearlines, and 'Origin All' (the default) will both print and scan the origins and tearlines.

Verbs:
    AREA
    NETAREA
    ECHOAREA

These three verbs control the actual searching for and extraction of messages. Due to the depth of this topic, they will be discussed in the next section.

Verb:
    SKIPBLANK

The SkipBlank verb controls how TopicX handles blank lines in messages. By default, TopicX will export the entire message, with or without blank lines. Some may want to use the SkipBlank option, since it disables the exporting of blank lines, thereby saving a few bytes in the ASCII extracts.

Verb:
    MACRO

The Macro keyword allows for definition of configuration-file macros, which can be substituted into any number of different places in the configuration file. Please note that macros must be defined before they are used, and they MUST be defined before the first Area/NetArea/EchoArea is used, or else an error will occur.

A macro definition has the following format:

Macro <macroname> <macro expansion>

<macroname> is the name of the macro to define, and <macro expansion> is the text to insert in its place when encountered in the configuration file.

Once a macro has been defined, it may be used anywhere in the configuration file, in place of normal text. TopicX will expand the macro before passing it through the date routines. (see "Including Dates in a TopicX Configuration File") To use a macro, enclose the case-insensitive macro name in "at" (@) signs.

Examples:

```
    Macro OutDir D:\Path
    Macro MessagePath E:\Msg
```

Given the above definitions, the following lines:

```
@outdir@\Myfile.Txt
@messagepath@\Meadow
```

would expand to:

```
D:\Path\Myfile.Txt
E:\Msg\Meadow
```

Verb:
    DIVIDER

The Divider verb specifies what to insert between each message extracted message, as a divider. The divider may consist of any characters, and may be up to 79 characters long. If no divider is specified, then TopicX will use a string of 79 dashes as a default.

Example:

```
  Divider ---= Brought to you compliments of Yogi ]I[ BBS =---
```

Verb:
    LOCAL

The Local verb controls TopicX's handling of local messages. Some of the more brain-dead Sysop message editors may not include proper date information inside the message, so they won't get scanned when using the -T or -Y command-line parameters. The "Local Redate" option specifies that you want TopicX to check the DOS datestamp on local messages, and scan the message according to that. However, this may not work with some message-base renumberers and reply-chain linkers, so this option may not be advisable. If your message base utilities are incompatible, the end result will be a local message being extracted each day TopicX is run, rather than just once. The "Local Import" option (the default) specifies that you want TopicX to try to import the ASCII datestamp contained inside the message, and use that as the message's date. If the ASCII date is invalid, then TopicX will check the binary datestamp which tells when the message was written. Failing that, TopicX will fall back to the "Local Redate" method. To disable this feature and use the standard datestamps (and thereby risk not scanning all locally-entered messages), use the "Local None" option. If TopicX doesn't seem to be scanning ANY messages with the date options, then please read through the Troubleshooting section.

Verb:
    MULTIPASS

The MultiPass verb is a command normally not needed, unless you are running with a very large configuration file. When using a large configuration file (more than about 20k), TopicX will quickly run out of memory in which to load

messages. The MultiPass verb specifies that TopicX is to make multiple passes through the configuration file, while keeping a certain amount of memory free for reading in messages.

The format for the MultiPass verb is:

```
MultiPass <messagemem>
```

<messagemem> specifies the amount of memory to save for holding the message body, in bytes. (A value from 10000 to 20000 is ideal)

The rest of available memory will be used for storing the configuration file information. When executing, TopicX will state which pass it is on, and the approximate amount of memory available for holding messages.

AREA DEFINITIONS

This is the bulk of the TopicX extraction system. Using the three area verbs (AREA, ECHOAREA, and NETAREA), you can configure TopicX to specify how to search for messages, where to search for messages, and what to do with them.

The general format of an Area statement is as follows (split in two to fit inside this document):

```
[Area│EchoArea│NetArea] <path> [desc] [<low water mark>
  [<high water mark>]]
```

There can be up to 512 Area/EchoArea/NetArea statements in a single configuration file. If you need more than this, either split the configuration file in two, or contact the author for an expanded version. Following the area line is a sequence of topic lines, which tells TopicX what to search for. An area definition runs from the first Area/EchoArea/NetArea keyword, UNTIL THE NEXT BLANK LINE, or until end-of-file. (This is important! If you do not insert a blank line between areas, TopicX will try to process the next Area verb as part of the previous area!)

The three different area verbs are used to specify which type of area TopicX is to search:

AREA - A normal, local message area. All messages are processed, with no special maneuvers on TopicX's part.

NETAREA - Process a NetMail area. This is the same as a normal area, except TopicX will add the origin/destination addresses to any messages extracted from this area.

ECHOAREA - Same as a normal area, except TopicX will skip processing the high water marker (1.MSG) if it

exists.

<path> is the path to the message area, with or without a trailing backslash. An example path would be "C:\Msg\Meadow", or "C:\Msg\Private\".

[desc] is an optional description of the message area, which will be included in the ASCII extracts. It MUST be enclosed in quotes, and it may be up to 30 characters long. Descriptions can be used with low and high water marks, as long as you insert the description BETWEEN the area path and water marks.

Low/High Water Marks

The low and high water marks are optional, and if specified, determine how many messages must be in each area to begin processing (low water mark), and the highest message number which will be processed by TopicX (high water mark). If you wish to process all messages in each area, then simply omit one or both of these parameters. The term "Low Water Mark" is a misnomer, since TopicX only uses that as a conditional before beginning to scan the area. If the required number of messages exists, TopicX will start scanning from the first message in the area, regardless of the low water mark. The low water mark must directly follow the pathname, separated by a space. If you wish to use a high water mark but not a low water mark, then set the low water mark to zero. The high water mark directly follows the low water mark, and may be omitted or set to zero to process all messages in the area. Note that the High Water Mark has no effect on the -T or -Y command-line parameters, and messages under the high water mark may not be processed due to their date, in spite of the High Water Mark.

But please be aware that the -F parameter DOES affect the High Water Mark; The -F parameter will then allow TopicX to run until it has processed and actually SEARCHED the number of messages specified by the High Water Mark. This differs from -T and -Y, since -T and -Y will only process up to the high water mark, whether or not any of the messages below get searched.

Examples:

The following command would process the EchoMail area in F:\Msg\Dr_Debug, but only if there were at least 50 messages in the area, and would only process the first 100 messages:

```
EchoArea F:\Msg\Dr_Debug 50 100
```

The following would process the NetMail area in C:\Netmail, but would stop processing when it reached the 50th message:

```
NetArea C:\Netmail 0 50
```

The following would process all of the messages in the local message area D:\Msg\Local:

```
Area D:\Msg\Local
```

TOPICS

What follows the area declaration are "topics", one to a line. There can be a maximum of 100 topics in each area, and there must be NO blank lines between topics. If you have more than 100 topics in one area, either contact the author for an expanded version of TopicX, or split the topics into groups and make two passes of the same area.

A topic is composed of a group of extraction flags, keywords to search for, binding operators between the keywords, and an ASCII output and optional archive path.

Specifically, a topic has the following format: (Split into two lines so it could fit into this document)

```
<[<flag>...][[&||][attributes...]...]> [[[<keyword> <operator>]...]
        <keyword>] [<output path> [archive path]]
```

Flag Groups

[<flag>...] specifies a "flag group", which must be the first "word" on each topic line. Flag groups control the extraction and searching operations of TopicX. A flag group is composed of a group of individual characters, each representing an individual flag. The flag group must be a single "word" (with no spaces between flags), however most flags can be specified in any combination, and in any order. Each flag designates an option TopicX is to perform on each message in the area. TopicX will accept any of the following flags:

T Specifies that TopicX is to search the to/from fields of each message. TopicX will prefix the words "To: " and "From: " to each message header, so you can have TopicX only pick up messages FROM a certain person by including the word "From: " in your search text, or search for messages TO a certain person by including the word "To: " in your search text.

S Specifies that TopicX is to search the Subject field of each message. TopicX will prefix the word "Subj: " before the subject field, so it is also possible to search only on a specific subject. Prefixing "To: ", "From: ", or "Subj: " to each message header is normally only useful if you're using a combination of to/from, subject, and message extracting.

M Search the message body of each message. Please note that TopicX will only search the parts of the message you specify, so to search the entire message and headers, you need to specify all three options. (To/From, Subject, and Message)

F FloobyDust: All messages in this topic will be sent to the extraction file, regardless of keywords it may contain. If the FloobyDust flag is present, then no keywords are allowed on the topic line, and the file to extract to (and optionally the archive name) must be placed right after the flag group.

L Log each message found under this topic. Using this for all message types will take lots of time (and disk space), but it's handy if you want to check for any "bad" words, or messages containing special keywords in your message base.

D Usually used in combination with the log option. TopicX doubles as a message-deletion system, so the 'D' flag causes TopicX to delete the message when it is finished processing the message. This should only be normally used with "bad" words, or for those dweebs that insist on leaving obscene messages. Since TopicX deletes the message only when it is finished processing, you can use the 'D' flag on any topic line, whether it falls first or last in the area.

A Archive the message text when finished extracting. This option will cause TopicX to invoke the archiver when all extraction has finished, and to insert the extract into the archive specified after the output path.

X Specifies that you want TopicX to skip to the next message if it encounters a match on this topic. Without this flag, TopicX will extract the current message to each topic a match is found in. If you don't wish this to happen, include the 'X' flag in the flag group for each topic in the area.

N This flag tells TopicX not to extract the current message. Any other options specified (logging, deleting, etc) will be performed, but the message will not be extracted. Make sure NOT to include an output path for this topic. Also, this parameter cannot be used with the archiving flag, for obvious reasons.

R Specifies that the keywords specified in this topic make use of UNIX-style regular expressions. Without this flag, keywords will be taken literally, and interpreted as-is. Due to the depth of this topic, is is covered in a separate section, "Regular Expressions". Please note that just specifying the regular expression flag can cause a decrease in search speed, so use this option only when necessary. If possible, specify the regular expression on another

topic line, using the same output file as the rest of the topic. (See below in the Advanced Techniques section)

Message Attribute Control

The message flags can optionally be followed by a set of attribute flags, which limit extraction based on each message's attributes. (Some examples of attributes would be CrashMail, Kill/Sent, Local, Private, etc) There are two distinct modes of attribute control -- The AND mode, and the OR mode. Using the AND mode, all of the specified attribute conditions must be met in order to search (and possibly extract) a message. Using the OR mode, one or more of the specified attribute conditions must be met in order to extract a message. To use the AND mode, append an ampersand ("&") to the flag group. To use the OR mode, instead append a pipe symbol ("|") to the flag group. After the AND/OR specifier, the flag characters take on different meanings. Also, a tilde ("˜") preceding a message attribute reverses the meaning of the attribute.

Example:

If &X extracts only messages with a certain attribute set, then &˜X extracts only messages WITHOUT that attribute set.

The message attributes which TopicX understands are:

P	Private
C	Crash
R	Read
S	Sent
F	File Attach
W	ForWarded
O	Orphan
K	Kill/Sent
L	Msg was entered locally
H	Hold
Z	File RequeZt
Q	Receipt ReQuest
T	ReceipT
A	Audit Trail Requested
U	Update Request

The two modes (AND/OR) may be combined at will, and may be used in any order. When using the two operators in conjunction, _all_ of the AND conditions must be met, and one or more of the OR conditions must be met. For example, "&CO|LS" will only process messages which have the Crashmail bit set, which are orphans, and are either local messages or have been sent.

Keywords and Operators

Following the flag group, comes the keywords, operators, output and archive paths. The keywords designate what TopicX is to search each message for -- They are also case-insensitive, so there's no need to search for both "John McDoe" and "John Mcdoe". There can be a maximum of 20 keywords per topic. If you need more than that number, either split the topic into two separate topics, using the same output path, or contact the author for an expanded version of TopicX.

By placing an operator between each keyword, you can create complex searches and extractions based on almost anything imaginable. The keywords must be enclosed in double quotes, and are each separated by an operator, which can be either an "&" (an AND), or an "|" (an OR). The AND operator will only extract messages in which all of the keywords can be found, and the OR operator will only extract messages in which one or more of the specified keywords can be found. To keep things simple, those are the only two operators allowed. (Future versions of TopicX may support more complex operators)

The AND operator has a higher precedence than the OR operator, so keywords using the AND operator bind "tighter" than the OR operator. For example, given the following expression:

```
"PkZip" & "Phil" | "Katz" & "Cats" | "PkArc"
```

The above would look something like this if it could be parenthesized:

```
("PkZip" & "Phil") | ("Katz" & "Cats") | "PkArc"
```

The keywords can also contain UNIX-style regular expressions, provided that the R flag was specified in the topic's flag group. (See the separate section on regular expressions)

If you wish to search for the double-quote character, a semi-colon (the configuration file comment character), or the literal of a standard regular expression character, please see the section regarding quoting.

Following the attributes (or following the flag group, if the FloobyDust flag was specified) is the output path. The output path is where the extracts of all qualifying messages get placed. If you specified the 'A' flag in that topic, following the output path should come the archive path, which is where the final archive will be placed. If you wish to have the extracts deleted after being inserted into the archive, then use the "Archiver" verb to tell your archiver to move extracts into the archives.

TopicX will only extract a message when it meets all of the conditions specified in the topic line. This means the message must have the correct date (if applicable), the

correct message attributes (if applicable), and the designated keywords (if applicable) must be found inside the message. If any of the above conditions is false, TopicX will move on to the next topic line, and eventually to the next message.

Order of Topics

When using specific flag groups and options, the order in which topics are listed becomes vital. TopicX scans the message starting with the first topic, then progresses to scan for the second topic, the third, etc. If you use the X (stop processing) flag, TopicX will stop processing the remaining topics, and skip on to the next message. This option is desirable if you don't want TopicX to extract to more than one output file, or if you want to screen out certain topics. For example, if you wish to extract certain topics from an area to only one output file, and to log other messages which weren't found by any of the other searches to a general file, you should arrange your topics in the following order:

```
XMS "Topic #1" Topic1.Out
XMS "Topic #2" Topic2.Out
XMS "Topic #3" Topic3.Out
XMS "Topic #4" Topic4.Out
F TopicAll.Out
```

Any of the above message flags can be changed, except for the stop and FloobyDust flags. The above insures that topics will only be extracted to one file, and only the topics which couldn't fit into one of the other files will be extracted to TopicAll.Out. If you don't want to extract messages which contained none of the keywords, then omit the FloobyDust topic.

REGULAR EXPRESSIONS

When the 'R' flag is specified in a flag group, TopicX will enable processing of UNIX-style regular expressions in keyword searches. Unless you specify the 'R' flag, each keyword will be processed literally, whether or not it contains regular expressions. And just as normal searches are case-insensitive, so are regular expression searches. The following characters can make up a regular expression:

* An expression followed by an asterisk wildcard matches zero or more occurrences of that expression[1]. For example, ba*h matches bh, bah, baah, baaah, but not bha.

+ An expression followed by a plus wildcard matches one or more occurrences of that expression[1]. For example, b+h matches bah and baah, but not bh.

^ Signifies the beginning of a line.

$ Signifies the end of a line. The '^' and '$' characters usually behave exactly the same (since the beginning of one line is usually the end of another), except at the beginning and end of messages and subjects.

[] A sequence of characters enclosed in brackets matches any character in that sequence. If the first character in the sequence is a circumflex, it will match only characters NOT found in the sequence. To specify a range of characters, use two characters separated by a dash. For example, "[a-gi-z<]" would match a left angle bracket, and any letter except h. "[^a-m]" will match letters N through Z. All of the normal regular expression characters are valid inside brackets, except for '.', '+', and '*'. Please note that there is NO difference between upper and lower case, even inside square brackets.

. Signifies that any character can be matched.

All other characters besides the above are processed literally, and must be found in the message text to constitute a match. To search for the literal of a regular expression character, prefix it with the backslash character.

---

[1] Currently, the '*' and '+' operators won't work in situations like these: "[abc]*b" or "abc.+x". If the character immediately following the '*' can be caught by the previous expression (the '[abc]' for the first, the '.' for the second), then TopicX will pretend the second part of the expression doesn't exist, and extract the message if the first part was matched. This should be fixed in an upcoming version]

Examples:

"colou*r"

The above would match both "color", "colour", "colouur", etc.

"gr[ea]y"

This expression would match both "grey" and "gray".

"[^f]iss"

The above would match the "hiss" or "hisses", but not "fissure".

"^well"

This would match "well", as long as it was at the at the beginning (or end -- See notes on the '$' character) of a line.

"[^blp]ack"

This would match "mack", but wouldn't match "back", "lack", or "pack" .

"^qu.ck\.*!"

The above would match "quack...!", "quack!", and "quick.!" if they were at the beginning of a line, but not "quicks!", "quacking." or "quack".

ADVANCED TECHNIQUES

Output Paths:

If you decide to specify the same output path more than once in each area, make sure that it is specified as being in the same directory, on the same physical drive. Since there it no rational way to determine if an output path has been duplicated, TopicX resorts to comparing the names of the output paths. TopicX will add drive specifiers and directories to ambiguous output paths, but it cannot compensate for referencing the same file twice, once as its real name, and once more on a different drive, such as could be obtained by using a SUBST command.

Regular Expressions:

Due to the loss of speed when processing regular expressions, it is sometimes beneficial to split a topic up so that only those keywords utilizing regular expressions use the regular expression searches. To do this, split the topic line into two separate lines (with identical paths), and insert all of the regular expressions on the first line. Then, remove the 'R' flag from the second line, add an 'X' flag to the first line, and you're all set.

For example, instead of using the following topic line:

```
RMS|RS "Gr[ae]y" | "Mauve" | "Brown" | "Beige*"
       D:\Path\File.Ext
```

You should use something like this for optimum speed:

```
XMS|RS "Mauve" | "Brown" D:\Path\File.Ext
RMS|RS "Gr[ae]y" | "Beige*" D:\Path\File.Ext
```

Word Delimiters:

Another really useful concept is to define a "word delimiter" macro. This is useful if you want to search for the name "Wes", but don't want to get "West" or "Northwestern":

```
Macro WD [^a-z0-9]

RSM "@wd@Wes@wd@" MyFile.Txt
```

Low/High Water Marks:

Low water marks are useful in situations where you don't want to use either of the date options (-T or -Y), but want to extract a given message only once. If you configure each topic in each area to delete messages after extraction (by specifying the 'D' flag for each topic, and specifying a 'FND' flag at the end of each area to nab messages not

extracted), you can extract only a certain number of messages each day and delete them as you go along, thereby alleviating the need for a separate message deletion utility. Since the message is deleted once extracted, there is no chance of it being processed twice. The low water mark was provided for situations when there is an irregularity in EchoMail flow: If there is a small number of messages in each base, you probably don't want to process and all of them. The low water mark CAN be set higher than the high-water mark, so it is possible to process up to the 50th message in an area, as long as more than 100 messages exist. See the section on Low/High Water Marks for more details.

EXAMPLES

Here are a few sample topic lines, to give you an idea of what TopicX can do:

```
T "From: Vince Perriello" | "From: Wynn Wagner" |
  "From: George Stanislav" OpusLCD.Txt
```

The above would search the To/From portions of each message, and extract each message from either of the above three individuals to the file OpusDev.Txt in the current directory.

```
AT|RS "From: John Doe" D:\Path\Johndoe.Txt D:\Arcs\Johndoe.Arc
```

The above topic would only process messages which had been received or sent, and from John Doe, to the file D:\Path\Johndoe.Txt. Once all message extraction is completed, TopicX would then invoke the specified archiver to either add or move Johndoe.Txt to D:\Arcs\Johndoe.Arc.

```
DLNM "Hello, world."
```

The above topic would create a log entry for each message found that contained the words "Hello, world" in the message body, and then delete that message.

```
AMS&RL "Subj: Fusion" | "Palmer" & "Stanley Pons" |
       "Deuterium" Coldfus.Txt Coldfus.Arc
```

The above topic would only extract messages which were written locally and received, and had the subject of "Fusion", or contained the words "Deuterium", or both "Palmer" and "Stanley Pons" in either the subject line or message body. The extracts would be output to the file Coldfus.Txt, and would be archived into Coldfus.Arc at the end of processing.

```
TD|RS&~Z~Q~T~A~O "To: Steven Bonisteel" File.Ext
```

The above would scan all messages which were either received or sent, and NOT a file request, NOT a receipt request, NOT a receipt, NOT an audit trail, and NOT an orphan. Of the messages actually scanned, TopicX would only extract messages TO Steven Bonisteel. The extracts would be contained in the file "File.Ext".

```
Macro WD [^a-z0-9]

EchoArea F:\Msg\Dr_Debug
  SM "Fat" | "Cluster Size" Clus%y%m.Txt
  SM "Fractal" Frac%y%m.Txt
LRSM "@wd@GNU@wd@" Gnu%y%m.Txt
  SM "Aspartame" | "Coke" | "Mountain Dew" | "Jolt" |
```

```
       "Pepsi" | "Dr Pepper" Soft%y%m.Txt
 RSM "Front *Door" Fdor%y%m.Txt
 RSM "@wd@BNU@wd@" Bnu%y%j.Txt
RXSM "[^\.]Zip" Zip%y%m.Txt
  SM "LHArc" Lhrc%y%m.Txt
```

The preceding area would:

1) Extract any messages containing the strings "Fat" or "Cluster Size" in the subject or message body, to the file Clus<year><month>.Txt, where <year> is the current year, and <month> is the current month.

2) Extract any messages containing the string "Fractal" to the file Frac<year><month>.Txt.

3) Extract any messages containing the WORD (ie. surrounded by non-alphanumerics) "GNU", and insert the extracts in the file Gnu<year><month>.Txt.

4) Extract any messages containing any of the following strings to the file Soft<year><month>.Txt: "Aspartame", "Coke", "Mountain Dew", "Jolt", "Pepsi", or "Dr Pepper".

5) Extract any messages containing the string "FrontDoor" or "Front Door" to the file Fdor<year><month>.Txt.

6) Extract any messages containing the WORD "BNU" to the file Bnu<year><juldate>.Txt, where <juldate> is the current julian date.

7) Extract any messages containing the string "Zip" to the file Zip<year><month>.Txt, as long as the string wasn't prefixed with a period, such as would occur in a filename.

8) Extract any messages containing the word "LHarc" to the file Lhrc<year><month>.Txt, as long as it hadn't been extracted to Zip<year><month>.Txt. (This is caused by the 'X' flag on the ZIP topic line)

For more examples, please refer to the enclosed configuration file, TOPICX.CFG.

TROUBLESHOOTING

If you encounter a problem when trying to use TopicX, then you should follow the following steps before trying to contact the author:

1) Make sure you've read the ENTIRE manual. Some of TopicX's options are very complicated, and you're sure to misunderstand if you haven't read over the manual very carefully.

2) Read the error message produced (if any). They will usually provide a clue to the solution. For example, if the error message states, "Error parsing line XX of configuration file", you should go over line XX (and the surrounding lines) with a fine-toothed comb.

3) If the error is in the configuration file, check over the problem lines carefully. If the error is in an area definition, then check the entire area for mistakes. Also, a list of error messages TopicX can produce follows at the end of this section. Check to see if the error message you received is listed there, and read the hints listed following the error message.

If TopicX encounters an error, it will return one of the following DOS errorlevels:

Errorlevel 0 - No Error.
Errorlevel 1 - An error occurred parsing the command-line.
Errorlevel 2 - An error occurred parsing the configuration file.
Errorlevel 3 - Ran out of memory.
Errorlevel 4 - Error in opening/reading/writing a file. (The disk could be full, or a needed file might not exist)
Errorlevel 5 - A bizarre internal logic error occurred.

A common error occurs when using an improperly-configured mail processor. If TopicX fails to extract messages when using either the -T or -Y options, that probably means the date field was corrupted by a mail-handling utility. Users of ConfMail and other processors should MAKE SURE that the datestamps in imported messages are being converted to the "new" Opus format. This means using the "-N" switch for ConfMail Import, and using the "Opus_Date" keyword for QuickMail. OOMP users need not worry about this, since Opus' internal tosser dates imported messages automatically. (Other processors should have similar options.)

If the above fails to resolve anything, run TopicX over the suspected area with the -D (date debug) command-line parameter. This will display the three different datestamps stored in each message scanned. The one TopicX is concerned about is the "date arrived" stamp -- If this stamp has been zeroed, you can be assured that TopicX won't work with either the -T or the -Y options. Once you've deduced this, try to narrow down the cause and determine which message-handling

utility on your system is causing this problem, and either fix or remove that utility.

If TopicX encounters an error while parsing the configuration file, it will display a message to the local console. Since TopicX can't insert this message in the log file (which must be specified in the configuration file), you should trap the errorlevel returned from TopicX, and add a message to the system log if an error occurred.

The current version of TopicX will display any of the following error messages while parsing the configuration file or scanning messages:

"Error! Invalid version of compiled config file
D:\Path\File.Ext!"

This means that you tried to use an old version of a compiled configuration file format. This won't normally happen, unless you're upgrading from a previous version of TopicX. If this happens, the solution is to delete the .DAT file and start again.

"Error in compiled config file `D:\Path\Config.Dat'!"

This means that the compiled configuration file is corrupted. Again, the solution is to delete the .DAT file.

"Error! Either configuration file line XX isn't terminated with a CR/LF pair, or the line was longer than 255 characters!"

This means either:

(a) You created the configuration file with a non-standard editor. Lines must be terminated with a carriage return and a linefeed, and must use the standard ASCII codes.

(b) The line was longer than 255 characters. If the line is a topic line, then split it into two different lines using the same output path.

"Error! Invalid <item> specified on line <line> of
configuration file"

This means that a particular part of a line could not be processed correctly. <item> can be one of the following:

verb - You specified a non-existent verb, or inserted a blank line in the middle of an area definition. You may have misspelled a verb (such as using "MultePass" instead of "MultiPass"), or forgotten to use a comment

character in front of a comment.

operator - You specified an operator other than '&' or '|'. Check the topic line to make sure everything is where it is supposed to be, and you haven't forgotten to use double quotes around the keywords. Also make sure you haven't specified an archive path when none was required.

keyword - You may have forgotten to include a keyword on a topic line (if you're not using the 'F' flag), or you may have forgotten a double-quote somewhere.

output path - You either forgot to specify an output path, or forgot to use the 'N' flag.

archive path - You either forgot to specify an archive path, or mistakenly used the 'A' flag.

"Error! Too many <items> specified on line <line> of configuration file! (Max <max>)"

You specified too many of a particular type of item in the configuration file. There is a limit on the number of macros, areas, topics, and keywords:

64 macros per configuration file,
512 areas per configuration file,
100 topics per area,
20 keywords per topic.

"Not enough memory to compile area <area> with given MultiPass value!"

This means that TopicX didn't have enough memory to compile the specified area using the given MultiPass value. TopicX must have enough memory to compile this area in memory. If the MultiPass value is set too high, there won't be enough room to compile even part of the area, so TopicX will abort. The solution is to adjust the MultiPass value downwards until there is enough memory to compile the area.

"Error! Message directory <path> does not exist!"

Each area you specify in the configuration file must exist. The solution is to create the area.

"Error in regular expression on line XX of configuration file! Unterminated [] character set!"

In one of the regular expressions on that line, you have a '[', but no corresponding ']'. To correct the problem, insert

a ']' in the proper place. If you wish to use a literal '[', then see the section on quoting. This situation may also occur if you use the '[' character in a normal keyword, and mistakenly use the 'R' flag.

"Error in regular expression on line XX of configuration file! You cannot specify an empty [] character set!"

This means that you must have something inside a character set. In other words, an expression such as "abc[]efg" is invalid.

"Error in regular expression on line XX of configuration file! There must be a valid character following the quote character ('\')!"

You tried to use the quote character as the last character in a regular expression. Since the quote character signifies that TopicX is to take the literal of the next character, it cannot be the last character in a keyword.

"Error in regular expression on line XX of configuration file! '*' or '+' cannot be the first character in the expression!"

This means that you tried to use either '*' or '+' as the first character in a regular expression. Since these characters represent a repeat of the preceding character, it cannot be used as the first character.

"Error in regular expression on line XX of configuration file! You cannot specify two repeat characters ('*' or '+') in a row!"

This means you tried to use two repeat characters in a row. Since a repeat character specifies a repeat of the preceding character, two repeat characters in a row is illegal.

"Error! Too many arguments on line XX of configuration file!"

This means you used too many options/words on a particular line in the configuration file. An example would be specifying the F (FloobyDust) flag, and specifying an extra keyword by mistake.

"Invalid flag `Y' in line XX of configuration file!"

This means that you either forgot to add a flag group to a topic line, or used an invalid flag in the flag group. You also may see this error message if you forget to insert a blank line between each area.

"Invalid attribute type `Y' in line XX of configuration file!"

This means that you specified an invalid message attribute after the flag group. Valid attributes are listed in the "Message Attribute Control" section of this document.

"Ran out of memory while parsing configuration file!"

This means that there wasn't enough memory to fully compile the configuration file. See notes on the MultiPass verb, which allows you to parse configuration files larger than memory allows.

"Error! No searching operation specified in flag group on line XX of configuration file!"

This means that you didn't specify either of the T, M, S, or F flags. You must either specify a searching operation, or the FloobyDust flag (process all messages) for TopicX to work correctly.

"Error! Scan types N and A cannot be mixed in line XX of configuration file!"

This means you tried to specify both the N and A flags in the same topic line. Since the N flag specifies that you want no output file, the A flag (which specifies archiving of output files) cannot be rationally used in the same topic.

"Error! <item> is too long in line XX of configuration file!"

This item occurs if a string is too long for TopicX to handle. TopicX has a fixed length for several options (such as the archiver name, message divider, and macro names), and will abort if you specify a string too long to fit in the allocated space.

"Ran out of memory while parsing configuration file!"

This means that there wasn't enough memory to read and use the entire configuration file in memory. See the 'MultiPass' verb for more information on how to get around this.

"Undefined macro `<macro>'"

This error occurs when you use an undefined macro in the configuration file. You may have forgotten to double-up a literal commercial at-sign (you should use "@@" instead of "@"), or you may have tried to use a macro before it had been defined.

"Error! Extraneous archive path specified on line XX of configuration file!"

This means that you specified an archive path where none was required. In other words, you used an archive path in a topic which didn't contain the 'A' flag.

"Error opening output file `file'!"

This means that TopicX couldn't open a specific output file, usually due to an invalid filename, or not enough directory entries left in the root directory. If the problem is in the path, then fix it and re-start TopicX.

"Bizarre error parsing compiled regular expression!"
"Bizarre scan-range!"

Either of these errors means that there was an internal logic error in TopicX, which can't be fixed by ordinary means.

REVISION HISTORY

1.00 Original release, after several months of alpha, beta, and gamma testing.

CREDITS

Many thanks go to Ken McVay and Steven Bonisteel, my all-in-one beta test team, without whom this program would not have been possible.

This document was written and produced by PC-Write v3.01, a Shareware word processor. If you'd like to learn how to use PC-Write to produce this kind of document, then by all means drop me a note at the address listed below.

NO WARRANTY

All software in this package, and accompanying written materials and documentation (including instructions for use) are provided "as is", without warranty of any kind. Furthermore, Scott Dudley does not warrant, guarantee, or make any representations regarding the use, or the results of use, of the software or written materials in terms of correctness, accuracy, reliability, currentness, or otherwise. The entire risk as to the results and performance of the software is assumed by you. The above is the only warranty of any kind, either express or implied, statutory or otherwise, including but not limited to the implied warranties of merchantability and fitness for a particular purpose that is made by the author on this product. In the words of Wynn Wagner III, "If you break it, you get to keep both parts."

###