checkaddr

MMDF address verification program

Syntax

/usr/mmdf/bin/checkaddr [**-w**] [***addresses...***]

Description

The **checkaddr** program is used to check the validity of an address within the local mail system (MMDF). **checkaddr** can be given addresses either on the command line, one address per argument, or a list of addresses can be given to **checkaddr** on the standard input, one address per line. The latter mode is used for checking the addresses in a mailing list as in **checkaddr <** ***mailing-list-file***. **checkaddr** announces each address on a separate line and follows the address with its status (normally "OK"). **checkaddr** uses **submit**(ADM) to do the address verification.

If the **-w** option is given, **checkaddr** causes **submit** to generate a detailed submission tracing. This can sometimes be useful to help find problems in alias files or mailing lists.

See also

submit(ADM)

Credit

MMDF was developed at the University of Delaware and is used with permission.

checkque

MMDF queue status report generator

Syntax

/usr/mmdf/bin/checkque [**-fpsz**] [**-t***age*[**m**]] [**-c** ***channel channel*** ...]

Description

checkque reports on the amount of mail waiting in the MMDF distribution queue. It indicates the total number of messages and the size of the queue directory. It then lists the number of messages waiting for each transmission channel.

The **-c** option allows one or more channel names to be specified. If present, **checkque** restricts its report to the named channels.

The **-f** option causes **checkque** to print the name of the oldest queued message for each channel. **-p** causes only channels with "problems" to be listed. Problems are defined as channels with mail waiting for over some "problem threshold". The default problem threshold is 24 hours. The **-t** option is used to change the problem threshold. A number of hours (or minutes, if **m** is appended) should appear without a space after the **-t**. **-s** forces an abbreviated summary listing instead of the normal multi-line report. **-z** causes channels with no messages queued to be skipped in the report.

Because the mail queue usually is protected from access by any uid, except MMDF, **checkque** should be run under *root* or *mmdf* uid. It should not be made **setuid**( ) to *mmdf* unless you want to allow non-staff members to see the queue status.

Most configurations will have only two channels. One is for local delivery and the second is for off-machine relaying, such as by calling out or by being called up, or by attaching to ArpaNet hosts. Local delivery usually happens at the time of submission, so it is rare that any mail is waiting in it. Mail in other outbound queues is processed by **deliver** according to your site parameters, either by running **deliver** as a background daemon or by periodically firing it up via **cron**.

Files

quedfldir[]*/addr*
quedfldir[]*/msg*
quedfldir[]*/q.**
phase-directory/channel/*

See also

deliver(ADM)

Credit

This utility was written by Dave Crocker, Dept. of E.E., Univ. of Delaware.

MMDF was developed at the University of Delaware and is used with permission.

cleanque

send warnings and return expired mail

Syntax

/usr/mmdf/bin/cleanque [-w]

Description

cleanque removes extraneous files from the *tmp* and *msg* subdirectories of the MMDF "home queue" directory. It also sends warnings for mail which has not been fully delivered after "warntime" hours following submission. Finally, it returns mail which has not been fully delivered after "failtime" hours after submission. "Warntime" and "failtime" are defined in the MMDF *mmdftailor*(F) file.

Generally, **cleanque** should be run by **cron**, once a day, but may be run at any time to free up space.

The optional argument, **-w**, can be used if you are running **cleanque** manually and want to see what the program is doing.

See also

deliver(ADM), **queue**(F)

Notes

cleanque does not currently remove extraneous files from the individual queues (*q.** subdirectories).

Credit

MMDF was developed at the University of Delaware and is used with permission.

cnvtmbox

convert XENIX-style mailboxes to MMDF format

Syntax

/usr/mmdf/bin/cnvtmbox [**-c** | **-o**] ***old_mailbox*** [***new_mailbox***]

Description

cnvtmbox converts a mailbox (***old_mailbox***) either from the XENIX-style (the older UNIX-style) format to MMDF format or from MMDF format to XENIX format. Generally, mailboxes in MMDF format use ⟨Ctrl⟩**A** to delimit messages; XENIX format uses lines beginning with "From<space>" to delimit between messages. (You can change the message-delimiter character using the **MMBXPREF** and **MMBXSUFF** keywords in the */usr/mmdf/mmdftailor* file. For more information, see the **mmdftailor**(F) manual page.)

If ***new_mailbox*** is specified, **cnvtmbox** places the converted mailbox in this folder: otherwise, this utility writes the converted mailbox to *stdout*.

The options to **cnvtmbox** are:

c Converts XENIX-style or mixed-format mailbox to MMDF (generally ⟨Ctrl⟩A-delimited) format. If no options are specified, **-c** is the default.

o Converts MMDF or mixed-format mailbox to XENIX-style (or old UNIX-style) format.

File

/usr/mmdf/bin/cnvtmbox

See Also

"Setting up electronic mail" in the *System Administrator's Guide*

Value Added

cnvtmbox is an extension of AT&T System V provided by The Santa Cruz Operation, Inc.

Credit

MMDF was developed at the University of Delaware and is used with permission.

dbmbuild

build the MMDF hashed database of alias and routing information

Syntax

/usr/mmdf/table/dbmbuild [**-nvdk**] [***database*** [***table*** ...]]

Description

dbmbuild reads the tables specified in the MMDF tailor file into a hashed database for use in quickly verifying addresses and efficiently assigning channels to submitted messages. Whenever you change MMDF alias or routing information in any way, you must rebuild the hashed database by logging in as **mmdf** and running **dbmbuild** from the */usr/mmdf/table* directory.

If no database file is specified, the default database *mmdfdbm* is used. If no table files are specified, all tables listed in the tailor file are used. In particular, three tables are read for each channel definition: the list of authorized sources, the list of authorized destinations, and the table of names/aliases for that channel. Also, the remaining tables (MTBL and MDMN) are read.

The options are:

n Create a new database. If this option is omitted, **dbmbuild** updates an existing database. If no options at all are specified, **-n** is assumed; however, if you give any options (even **-v**), you must specify the **-n** option if you want to create a new database.

v Run in verbose mode, displaying information during table processing.

d Run in debug mode, reporting everything that happens.

k Keep going. If a file is mentioned that does not exist, ignore it. This option might be an appropriate default at some sites.

Appropriate locks are placed on the database so that **dbmbuild** can safely be run while MMDF is in operation.

Files

/usr/mmdf/mmdftailor	
/usr/mmdf/table/alias.list	
/usr/mmdf/table/alias.user	
/usr/mmdf/table/.chn*	
/usr/mmdf/table/.dom*	
$(tbldbm).dir	database directory
$(tbldbm).pag	database pages
$(tbldbm).lck	database locking file
*$(tbldfldir)/**	various tables that form the database

See also

dbm(S), **mmdftailor**(F), **tables**(F), "Setting up electronic mail" in the *System Administrator's Guide*

Credit

MMDF was developed at the University of Delaware and is used with permission.

dbmedit

edit the mmdf database file

Syntax

/usr/bin/dbmedit [**-v**] [**-d***database*] [***cmd****...*]

Description

The **dbmedit** command lets you edit the **dbm**(S) database used by MMDF. Use this command for quick and simple changes to the database or with careful use of **setuid** programs to make controlled changes on behalf of users. For example, a **forwardmail** command (that you create) might use **dbmedit** to change a user's entry in the *dbm* database after changing the mail forwarding alias file.

The **-v** option may be used to get a verbose description of the program's activities.

The **-d** option may be used to specify an alternate database. The default is given by the **tbldbm** configuration variable or by the MDBM **/usr/mmdf/mmdftailor** variable.

If no arguments are given to **dbmedit**, then the program goes into an interactive mode, and prompts the user for each command. Otherwise the arguments are taken as one command.

Commands in **dbmedit** refer to keys, tables, and values. Tables (see **tables**(F)) are hashed into the database using **dbmbuild**(ADM). (Tables that refer to domain name servers are not part of the database.) The keys appear on the left side of the tables and the values on the right side. In general, only the first occurence of a value for a given key/table pair is significant. For example, the table entries:

table1:
⟨key1⟩: ***val1***
⟨key2⟩: ***val2***

table2:
⟨key1⟩: ***val3***
⟨key1⟩: ***val4***

get hashed into the following database entries:

⟨key1⟩ *table1* ***val1***
⟨key1⟩ *table2* ***val3***
⟨key1⟩ *table2* ***val4***
⟨key2⟩ *table1* ***val2***

(In the current implementation, the database is keyed on only the key, and table/value pairs are encoded in the data portion. This is likely to change but will not affect this or any other program.)

The command lines in interactive mode are parsed using the standard MMDF string-to-argument routines so the same quoting and escape conventions are used. For example, if you want double-quotes or spaces in the value, they must be escaped with a backslash or the string must be quoted (for spaces).

The commands are:

print ⟨key⟩ [***table***]
Print the value of the key/table pair. If the table is omitted, then print the value of any table entry with this key.

add ⟨key⟩ ***tablevalue***
Add a key/table entry with the given value. In verbose mode, a warning message is printed if the given key/table pair already has a value in the database.

delete ⟨key⟩*[table* [***value***]]
Delete the values for the specified key. If a table is specified, delete only the values for the specified key/table pair. If a value is also specified, delete only entries for the pair with that value. It is an error to try to delete something which does not appear in the database as specified.

change ⟨key⟩ *table* [***oldvalue***]***newvalue***
Change the value of the specified key/table pair to ***newvalue.*** If ***oldvalue*** is specified, change the entry matching that value. Otherwise, change the value of the first occurence or add a new key/table pair if none already exists.

help Give a brief summary of the commands

quit Exit the program.

All commands may be shortened to their first character only. If the wrong number of arguments is given to a command, a "Usage:" message is displayed. This program may be used while MMDF processes are running.

NOTE All changes are made in real time; no temporary copy of the database is made while editing takes place.

File

$(tbldbm).{dir,pag}	the MMDF database

See also

dbmbuild(ADM), **tables**(F)

Credit

This utility was written by Phil Cockcroft.

MMDF was developed at the University of Delaware and is used with permission.

deliver

MMDF mail delivery process

Syntax

/usr/mmdf/bin/deliver [**-bdpsw**] [**-c***chan,chan*] [**-l***mins*] [**-t***hrs*] [**-m***maxsort*] [**-L***logfile*] [**-T***secs*] [**-V***loglevel*] [***message1 ... messageN***]

Description

The **deliver** program handles the management of all mail delivery under the MMDF mail system. **deliver** does not deliver mail directly, but instead calls on MMDF channels to handle actual delivery. **deliver**'s actions are guided by the MMDF tailoring file, */usr/mmdf/mmdftailor*, and by the command line options. The program can run as either a daemon or a user-invoked program. The program may be called to process the entire mail queue or just handle some explicitly named messages. When possible, **deliver** will attempt to process messages in the order received. **deliver** also maintains a cache of host information on a per-channel basis which allows hosts which are unavailable for delivery to be skipped until available.

deliver first builds a list of channels to process, either from the command line or composed of all the non-passive channels in the system. Next, a list of messages to process is collected, either from the command line or by scanning the mail queue for each channel. If the the number of messages in the queue for a given channel is more than ***maxsort*** (set in the tailor file or on the command line), the queue directory for that channel will be processed in the order read, without sorting by submission time. If a list of messages is given on the command line, no sorting will take place and the messages will be delivered in the order specified. The sorting keys are (in order): **channel**, **submission time**, and finally **host**. This causes many accesses to the messages but minimizes the invocation of channel programs.

deliver is **setuid** to the super user to allow it to set its real and effective UID and GID to that of the MMDF user.

The following options may be used to alter **deliver**'s behavior:

-b Background mode. Causes **deliver** to run as a background daemon making periodic sweeps over the mail queues looking for undelivered mail and attempting deliver. The invoker must be the MMDF user or the superuser to use this option. **deliver** attempts delivery for all eligible messages, then sleeps, and then repeats the process. The default sleep time is 10 minutes but it can be changed (see the **-T** option below).

-c*channel1*,*channel2*,...
Channel selection. A comma-separated list of channels to be processed.

-d Already in "***quedfldir***". This option will cause **deliver** to assume it is already in the mail queue and therefore it will not issue an explicit **chdir** . This is useful if you wish to have **deliver** operate on an alternate mail queue hierarchy, mainly for testing.

-l*minutes* Sets the "time-to-live" for entries in the dead-host cache. This time defaults to 2 hours. The dead host cache is used to prevent attempts to deliver to hosts that are known to be down. The "time-to-live" is given in minutes. If the number of minutes is negative, dead host caching is disabled.

-m*maxsort* Sets the sort threshold. If there are more than ***maxsort*** messages in a given channel's queue, then they are processed in directory order without first sorting by submission time. If **-m** is not specified, the value of ***maxsort*** is given in the tailor file by **MMAXSORT**.

-p Pickup only mode. Indicates that the invoker would like to pickup a passive mail channel.

-s Force linear search of the mail queue. Normally **deliver** will deliver messages in the order they were received which seldom matches the order in the directory. This option is useful if the queue gets so large that **deliver** can no longer deal with sorting the queue in a reasonable time.

-t*hrs* Time limiting. This option prevents **deliver** from attempting to deliver messages which have been in the queue for more than ***hrs*** hours. For efficiency reasons, this option only applies when the queue is being sorted. If an explicit list of messages was given on the command line, if the **-s** option is in effect, or there are more messages than the maxsort threshold (see the **-m** option), then time limiting does not occur.

-w Watch the delivery. Causes **deliver** to print informative messages on the standard output as it is attempting delivery. This option is passed onto the channel programs which also give informative messages.

-L*logfile* Sets the logfile for this **deliver** to the file specified. The default is to log into the file *msg.log* in the MMDF log directory. This option is only available to the superuser and MMDF.

-T*seconds* Sets the sleep time between background sweeps of the mail queue. This defaults to 10 seconds.

-V*loglevel* Sets the logging level for this **deliver** to the level specified. The ***loglevel*** should be a valid MMDF logging level string such as FTR. This option is only available to the superuser and MMDF.

See also

submit(ADM), **queue**(F), **mmdftailor**(F)

Value added

deliver is an extension of AT&T System V provided by The Santa Cruz Operation, Inc.

Credit

MMDF was developed at the University of Delaware and is used with permission.

list

list processor channel for MMDF

Syntax

list

Description

list is an MMDF channel program for handling mailing lists. The channel functions as a feed-through between **deliver** and **submit**. The list channel has its own host table and domain table with one entry for the pseudo host "list-processor" or something similar. This program is called by the program **deliver** and is not meant to be invoked by users directly.

The **list** channel performs two basic services. First, it postpones the verification of the list addresses and performs the (possibly lengthy) verification in the background when the **list** channel resubmits the message to the mail system. This prevents tying up a network connection or a user's terminal when verifying a long mailing list. Second, the **list** channel will, under special circumstances, change the return address for the message to a generic maintainer's address. The return address is determined by first taking the destination address (for example, "largelist") and seeing if there is an address in the alias file called "largelist-request". If there is, then "largelist-request" is used as the return address. If that was not found, the list channel checks to see if the destination address has a trailing "-outbound". If so, this is stripped and a "-request" is added and the lookup in the alias file is made a second time. If the "-request" address is found, then that address is used as the return address. If no "-request" address is found, then the original return address is used (normally the address of the sender).

To use the **list** channel to process a list, it is generally necessary to make three entries in the alias file(s). Let us say that we wish to set up a list called "largelist" and we want this list to be processed by the **list** channel. We would need the following entries in the alias file:

```
largelist:              largelist-outbound@list-processor
largelist-outbound:     :include: /usr/mmdf/lists/largelist-file
largelist-request:      maintainer
```

The first line causes mail sent to "largelist" to be sent through the list processor, readdressed to "largelist-outbound". The second line is what actually references the mailing list file for "largelist". The third line is optional, and is used to set up the (informal) standard maintenance address. This -request address, if present, will also be used by the **list** channel as the return address for mail submitted to the list.

See also

deliver(ADM), **submit**(ADM)

Files

<mmdf-table-directory>*/aliases* - to find list-request addresses

Credit

MMDF was developed at the University of Delaware and is used with permission.

mmdf

route mail locally and over any supported network

Description

The operating system uses MMDF (the Multi-channel Memorandum Distribution Facility) to route mail locally and over Micnet, UUCP, or other networks that provide MMDF support. The **custom**(ADM) utility installs MMDF and configures a basic system for sending mail on a local machine.

MMDF is a very versatile and configurable mail routing system. MMDF configuration begins with the */usr/mmdf/mmdftailor* file, which defines the machine and domain names, the various tables (alias, domain, channel), and other configuration information. To change the configuration of MMDF on your system, you can log in as *mmdf* and edit the configuration files. Whenever you change MMDF alias or routing information in any way, you must rebuild the hashed database.

Files

/usr/mmdf/mmdftailor
/usr/mmdf/table/alias.list
/usr/mmdf/table/alias.user
/usr/mmdf/table/.chn*
/usr/mmdf/table/.dom*
*/usr/spool/mail/**
/usr/spool/mmdf/...

See also

dbmbuild(ADM), **deliver**(ADM), **mmdfalias**(ADM), **mmdftailor**(F), **mnlist**(ADM), **submt**(ADM), **tables**(F), **uulist**(ADM)

"Setting up electronic mail" in the *System Administrator's Guide*

Value added

mmdf is an extension of AT&T System V provided by The Santa Cruz Operation, Inc.

Credit

MMDF was developed at the University of Delaware and is used with permission.

mmdfalias

convert XENIX-style aliases file to MMDF format

Syntax

/usr/mmdf/table/tools/mmdfalias

Description

mmdfalias is a conversion utility to produce MMDF-compatible alias files from the XENIX-format aliases file. **mmdfalias** also splits the converted contents of */usr/lib/mail/aliases* into two MMDF files containing list-type aliases and aliases that map users to machines.

After installing MMDF with **custom**, restore */usr/lib/mail/aliases* from backup tape. Place the following line in the file to indicate where the list aliases end and the mapping aliases begin:

```
# user-to-machine mapping
```

Log in as *mmdf* and run the **/usr/mmdf/table/tools/mmdfalias** conversion script from the */usr/mmdf/table* directory. You now have two MMDF files, *alias.list* and *alias.user*, in the current directory.

After creating these files in */usr/mmdf/table*, you must rebuild the MMDF hashed database. While logged in as *mmdf*, run **dbmbuild** from */usr/mmdf/table*.

Files

/usr/lib/mail/aliases
/usr/mmdf/table/alias.list
/usr/mmdf/table/alias.user

See also

dbmbuild(ADM), **tables**(F)

"Setting Up Electronic Mail" in the *System Administrator's Guide*

Value added

mmdfalias is an extension of AT&T System V provided by The Santa Cruz Operation, Inc.

submit

MMDF mail queue manager

Syntax

/usr/mmdf/bin/submit [-L...*V...*Wbcdf...*g...*hi...*jk...*lmnqrstuvwxyz]

Description

All mail is entered into the MMDF mail transport environment through the **submit** program. This document is intended to provide the specific information needed to control **submit**. While it can be called directly from a user's terminal, access to **submit** is most conveniently performed through a program such as **mail**(C).

Basic modes

submit permits considerable flexibility with respect to batching multiple submissions, response and error handling, and address source specification.

Multiple submissions

1. Terminate after one submission, such as is carried out by the mail command, or

2. Permit multiple message submissions, as is done by the SMTP channel.

The first mode is specified by passing any initialization information in the submit invocation line (that is, the **exec**(S) call). In the second mode, the initialization information is given as the first input line, for each submission. The format of this information is the same for both modes.

Response & error handling

1. Accept input until error or end of message, but terminate on any error, or

2. Notify result for each ***segment*** and continue.

Response mode #1 is mandatory with Multiple mode #1. Response mode #2 is called "protocol mode". During it, each address produces a status reply and the message text produces a reply. The domain of the term ***segment*** depends on the error. Simple addressing errors cause rejection only of the erroneous address. Other errors may cause rejection of the entire message, but permit submission of following messages.

Addresses

1. Extracted from components of the message text,

2. Explicit list given, ahead of message text, or

3. Both of the above (extracted and explicit addresses)

The first mode is common when mode #1 (non-protocol) is also in force for the Interaction and the Verification option. The second mode is commonly in force when the second modes apply for the other options (protocol mode).

Initialization

A message's initialization information is specified through a single string, passed either in the process-invocation argument list or in the first line of **submit** input. Hence, the string may be terminated either by a null or newline. Spaces and tabs in the line are ignored, unless part of a literal. Specification is only required for non-defaults.

	Option	Value	Literal
1.	Relay source for the "Via" or "Received" field	a. none b. source channel c. source host	(default) i...* h...*
2.	From/Sender authentication	a. reject on error b. trust c. no trust (disclaim)	(default) t u
3.	"Source-Info" field	a. not included b. disclaim author c. user text	(default) u f...*
4.	Address list source	a. explicit list b. extract from components c. both (extract and explicit)	(default) g...*
5.	Address verification	a. abort on invalid b. report on each address	(default) v
6.	Delivery destination	a. mailbox b. user's tty c. mailbox and tty	m (default) y b
7.	Delivery attempt (combinable)	a. leave for daemon b. deliver local now c. deliver netmail now	(default) l n

(Continued on next page)

(Continued)

	Option	Value	Literal
8.	Observation of immediate attempts	a. none b. user will watch	(default) w
9.	Return address	a. send to submittor b. send to "Sender:" c. do not return d. as specified	r s q (next line)
10.	Returned mail contents	a. entire original b. citation only	(default) c
11.	Warnings	a. send warnings b. do not send warnings	(default) z
12.	Delay channel usage	a. enable delay channel b. don't use delay	(default) d
13.	Delay channel indicator	a. not delay channel b. delay channel	(default) j
14.	Nameserver timeouts	a. short timeouts b. as specified	(default) k...*
15.	Submission tracing	a. not shown b. watch submission	(default) W
16.	Logging file	a. as per msglog b. as specified	(default) L...*
17.	Logging level	a. as per msglog b. as specified	(default) V...*

Comments

General:

Literals shown as characters, followed by an ellipsis, followed by an asterisk (for example x...*), represent a string. The first character specifies the nature of the setting. The value for the setting is placed between that character and the asterisk. The value may be any string not containing an asterisk, null, or newline. The values for settings **x** and **g** are comma-separated lists of strings. These strings may not contain asterisks, nulls, newlines, or commas.

Specific:

1. Relaying
   This is used when the calling program is interfacing with another distribution system, effecting relaying. The literal after the **i** specifies the channel the message is coming from. The **h** may be used, in conjunction with **i**, to specify the source host. The literal is the name of the host.

2. From/Sender authentication
   Normally, the message must correctly identify its sender. Anyone may send "anonymous" (unsigned) mail, but they must use the **u** setting which bypasses authentication. However, it also causes MMDF to include, in the "Source-Info:" component, a statement noting the absence of authentication. Only *root* or relays may use the **t** setting, which bypasses authentication and does not add a disclaimer. Others requesting it get **u** treatment.

3. Source-Info field
   In addition to the action explained above, "Source-Info:" can directly receive text, from the user, through the **f** setting. The value string is replicated on a separate line in the field.

4. Address list source
   An explicit list has one address per line. When **x** or **g** are specified, they list the names of message components, such as "To:" and "CC:", which are to be searched for addresses.

5. Address verification
   Normally, any illegal address will cause the entire message to be rejected. In **v** (verify) mode, the acceptability of each message is reported and encountering an illegal address does not abort submission.

6. Delivery destination
   Mail may be delivered to a recipient's *mailbox* (file), online terminal (if the recipient is logged in), or a combination of the two. There is no default. For each message, its delivery mode must be specified.

7. Delivery attempt
   An immediate attempt causes a special **deliver** process to be forked and it will attempt to process the indicated mail immediately. (The **n** setting does not allow more granularity, for historical reasons.) Otherwise, the system's background daemon will get to it eventually. The daemon also handles mail that initially could not be delivered/relayed. A channel's descriptor structure (in *chan.c* or the runtime tailor file) specifies a channel as being Active, Passive, or Background. Only the first is processed by any request for immediate delivery. The second indicates a Post Office Box-style channel. The third limits the channel to processing by the background **deliver** daemon, which may be necessary for restricting access to special channels, such as dial-out telephones.

8. Observation
   If an immediate attempt is requested, the user may elect to watch its progress. **deliver** and its children will report assorted aspects of their activity. If a quiet attempt is requested, **submit** returns as soon as submission is completed. That is, a quiet attempt is performed detached.

9. Return address
   If the invoker of **submit** is not to receive return mail (e.g., notification of delivery failure) then the next input line (the first, if settings are specified in the **exec**(S) call), contains an address that should receive the notification. It is not validated. If either the **r** or the **s** switch is given, **submit** will not read a line for the return address. If no return mail should be sent, the return address line should be empty (i.e., consist of a newline, only.) If the **q** switch is given, a return address is read from the next line of input but the local system will not return mail if delivery problems are encountered. The return address given may be used by other systems (if there are mail relays between the local system and the recipient).

10. Returned mail contents
    Normally, a copy of the entire message is sent with a delivery-failure notice. Using the **c** switch causes a citation, comprising the message header and first three lines of non-blank lines of the body, to be sent. If more than 12 addresses are specified, for a message, citation-only is automatically set. In addition, no warning message will be sent for addresses which take a long time to process (a site dependent value); the final failure notice will always be sent, if there are addresses that are never fully processed.

11. Warnings
    Normally MMDF will send a non-delivery warning if a message has been undelivered after a small period (typically 12 to 72 hours, depending on the site). Deliver attempts continue until a timeout period is reached. This is typically after 3 to 10 days, depending on the site.

12. Delay channel usage
    The delay channel is used to process mail submissions that could not be queued because necessary nameserver information was unavailable and therefore an authoritative decision on the validity of the address was not possible. If the **d** option is specified, use of the delay channel is prohibited. If the nameserver fails, an error is returned, rather than a conditional **OK**.

13. Delay channel indicator
    This option is intended only to be used by the delay channel itself to indicate to **submit** that the invoking process *is* the delay channel. This option implies the **d** option above.

14. Nameserver timeouts
    By default, MMDF uses a short timeout algorithm. This is suitable for user interface programs which do not want to wait a long time for dead nameservers. The **k** option allows a different timeout to be set. The value given is the number of seconds to wait for the nameserver lookup to complete.

15. Submission tracing
    The **W** option causes submit to print a detailed description of its activities on file descriptor 2. It will indicate, for each addressee, the channel and addresses queued. This can generate a great deal of output if a mailing list is encountered, so it should be used with caution.

16. Logging file
    The **L** option allows the specification of an alternate logging file at runtime. The string following the **L** should be the name of the logfile to be used. It can be terminated by a * or the end of the arguments. This option is only available to the super user or MMDF.

17. Logging level
    The **V** option allows the setting of the logging level at runtime. The string following the **V** should be one of the valid MMDF logging level strings such as **FTR** or **BST**. It can be terminated by a * or the end of the arguments. This option is only available to the super user or MMDF.

Input stream

The following augmented BNF characterizes **submit**'s input (file descriptor zero) format:

stream:	*(init-seq '\n' msg-info null) [null]
init-seq:	*{ switches listed above }
msg-info:	[ret-addr] '\n' [addr-seq '!' '\n'] { rfc822-format message }
ret-addr:	{ rfc822-format (return) address }
addr-seq:	*{ rfc822-address }

Address format

Addresses are expected to conform to the ARPANET mail standard known as RFC-822, available from the Network Information Center at SRI International. **submit** (and MMDF in general) also continues to support RFC-733 style mail for compatibility with earlier mail systems.

In addition to those in RFC-822, the following address delimiters are recognized within the local part of addresses (in order of precedence):

@
%
!
.

The "!" delimiter is interpreted as "host!user" while the others are interpreted as "user?host". For example, the address "a.b!user%c@localhost" would be queued for "a.b!user@c". The address "a.b!user@localhost" would be queued for "user@a.b". The address "user.a@localhost" would be queued for "user@a". Note that recognition of the "." delimiter is a site-selectable option.

Also, addresses may be indirectly referenced, through a file specification of the form:

"**<*filename*** or "**:*include*:*filename***"

where the angle-bracket must be the first non-blank character of the specification (to distinguish it from the "<...>" usage, above).

Addresses in the file may be separated by commas or newlines.

Example interactions

Phases involve Invocation (Invoke), data sent into **submit** via its file descriptor zero (To), data returned from **submit** via its file descriptor one (From), iteration back to the specified phase (Loop), and process exit value (Exit).

1. Simple, single-message command:

   a. Invoke: Parameters, "-mlrxto,cc*", indicate that the message is to be sent to recipients' mailboxes, local mail should be sent immediately, return mail goes to the submittor, and addresses are to be extracted from the "To:" and "cc:" components.

   b. To: The entire message

   c. From: Error messages

   d. Exit: Process return value, in wait(&val), indicating submission status.

2. Standard, multi-message protocol:

   a. Invoke: No parameters

   b. To: Initialization information line. A typical user program might have "mlrv", indicating the message is to be sent to mailboxes, local mail sent immediately, return mail goes to the sender, and each address verification is to be reported. A relay program might have "mlntviVGR.BRL.MIL*," with "mlv" as above and the other settings indicating that mail for non-local channels is to be sent immediately, the author information is to be trusted, and the "Received:" component should cite the mail as being relayed via Internet host VGR.BRL.MIL.

   c. To: One address, terminated by a newline ('\n').

   d. From: Status character, from *mmdf.h*, plus human-oriented text plus newline.

   e. Loop: Back to (c). Terminate with address line having only an exclamation mark (**!**), with newline.

   f. To: Message text, in Internet RFC #822 format. Multi-line, terminated by null ('\0').

   g. From: Status character, text, newline.

   h. Loop: Back to (b). Terminate with initialization line having only a null, without newline.

Channels

When MMDF is used in conjunction with the DARPA domain nameserver system, a "delay" channel should be configured to allow queuing of addresses that fail verification temporarily due to nameserver failures (unavailability). Two other special channels that can be configured are the "badusers" and "badhosts" channels. Mail to unknown users or unknown hosts will be queued to these channels if they are configured. The bad channels have no special code associated with them. The channel configuration should reference whatever table and parogram is necessary to reach a smarter host which can deliver or forward the mail. The channel should have the "host=" parameter set to this host name. The channel names given above are reserved.

Files

Numerous. Generally under the MMDF login directory.

See also

deliver(ADM), **mmdf**(S)

Credit

MMDF was developed at the University of Delaware and is used with permission.