# logs

MMDF log files: system status, error, and statistics logging for MMDF

## *Description*

MMDF maintains runtime log files at several levels of activity. The primary distinction is among message-level, channel-level, and link-level information. All logging settings can be overridden by entries in the runtime tailor file . In MMDF, that member is merged with */usr/mmdf/log* to determine the full pathname to the log. Logs are protected so that any process may write into them, but only MMDF may read them (that is, 0622).

The logging files may be the source of some confusion, since the log package entails some complexity. Its three critical factors are coordinated access, restricted file length, and restricted verbosity.

The length of a logging file can be limited to 25-block units. This is extremely important since files can grow very long over a period of time, especially if there are many long messages sent or very verbose logging.

Restricted verbosity is a way of easily tuning the amount of text entered into the log. This is probably the one parameter you need to be most concerned about. Set to full tilt (level=FTR), MMDF becomes noticeably slower and I/O bound. It also shows what it is doing, and helps you to discern the source of errors. When you are used to the code, setting the logging level down is highly recommended. The lowest would be TMP or FAT, for temporary or fatal errors. GEN will log errors and general information. FST logs errors, general and statistics information.

### *Specific logs*

Even with the listed divisions, the logs contain a variety of information. Only the message-level log's format will be explained in significant detail.

msg.log — records enqueue and dequeue transitions, by **submit** and **deliver**. Entries by a background **deliver** process are noted with a "BG-xxxx" tag, where the x's contain the 4 least-significant decimal digits of the daemon's process id. This is to allow distinguishing different daemons. When **deliver** is invoked, by **submit,** for an immediate attempt, the tag begins with "DL" rather than "BG". Entries by **submit** begin with "SB".

Every major entry will indicate the name of the message involved. Entries from **submit** will show "lin" if the submission is from a user on the local machine. In this case, the end of the entry will show the login name of the sender. If the entry is labelled "rin," then the mail is being relayed. The channel name, source host, and sender address are shown. Within parentheses, the number of addressees and the byte-length of the message are listed.

Entries from **deliver** show final disposition of a message/addressee. These are indicated by "end." Then, there is the destination channel and mailbox name. In brackets, the queue latency for the address is shown in hours, seconds, and minutes.

chan.log — records activity by the channel programs, in *chndfldir[]*. Entries have a tag indicating the type of channel making the entry. Different channels record different sorts of information. For example, the local channel shows when a **rcvmail** private reception program is invoked.

## *See also*

**mmdf**(ADM)

## *Value added*

**logs** 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.

# maildelivery

user delivery specification file

## *Description*

The delivery of mail by the local channel can run through various courses, including using a user tailorable file. The delivery follows the following strategy, giving up at any point it considers the message delivered.

1) If the address indicates a pipe or file default, then that is carried out.

2) The file *.maildelivery* in the home directory is read if it exists and the actions in it are followed.

3) If the message still hasn't been delivered, then it is put into the user's normal mailbox.

The format of the *.maildelivery* file is

***field*** **<FS>** ***pattern*** **<FS>** ***action*** **<FS>** ***result*** **<FS>** ***string***

where:

***field*** is name of a field that is to be searched for a pattern. This is any header field that you might find in a message. The most commonly used headers are: From, To, cc, Subject and Sender. As well as the standard headers, there are some pseudo-headers that can also be used. These are:

**source**
the out-of-band sender information. This is the address MMDF would use for reporting delivery problems with the message.

**addr**
the address that was used to mail to you, normally yourname or yourname=string (see below).

**default**
if the message hasn't been delivered yet, this field is matched.

***** this case is always true regardless of any other action.

***pattern*** is some sequence of characters that may be matched in the above ***field***. Case is not significant.

***action*** is one of the mail delivery actions supported by the local channel. Currently the supported actions are **file** or **>**, which appends the message to the given file, with delimiters; **pipe** or **|**, which starts up a process with the message as the standard input; and **destroy** which throws the message away.

For piped commands, the exit status of the command is significant. An exit status of 0 implies that the command succeeded and everything went well. An exit status of octal 0300-0377 indicates that a permanent failure occurred and the message should be rejected. Any other exit status indicates a temporary failure and the delivery attempt will be aborted and restarted at a later time.

***result*** is one of the letters A, R or ? which stand for Accept, Reject and "Accept if not delivered yet". They have the following effects:

**A** If the result of this line's action is OK, then the message can be considered delivered.

**R** The message is not to be considered delivered by this action.

**?** This is equivalent to **A** except that the action is not carried out if the message has already been accepted.

The file is always read completely so that several matches can be made, and several actions taken. As a security check, the *.maildelivery* file must be owned by either the user or root, and must not have group or general write permission. In addition the system delivery file has the above restrictions but must also be owned by root. If the field specified does not need a pattern, a dash "-", or similar symbol is usually inserted to show that the field is present but not used. The field separator character can be a tab, space or comma ",". These characters can be included in a string by quoting them with double quotes ("); double quotes can be included preceded by a backslash "\".

MMDF treats local addresses which contain an equal sign "=" in a special manner. Everything in a local address from an equal sign to the @ is ignored and passed on to the local channel. The local channel will make the entire string available for matching against the "addr" string of the file. For example, if you were to subscribe to a digest as "foo=digest@bar.NET", **submit**(ADM) and the local channel will verify that it is legal to deliver to "foo", but then the entire string "foo=digest" will be available for string matching against the MMDF file for the "addr" field.

## *Environment*

The environment in which piped programs are run contains a few standard features, specifically:

**HOME** is set to the user's home directory.
**USER** is set to the user's login name.
**SHELL** is set to the user's login shell (defaults to */bin/sh*).

The default umask is set to 077 (a very protective creation mask). A shell script can be run first to set up more complex environments.

There are certain built-in variables that you can give to a piped program. These are **$(sender)**, **$(address)**, **$(size)**, **$(reply-to)** and **$(info)**. **$(sender)** is set to the return address for the message. **$(address)** is set to the address that

was used to mail to you, normally 'yourname' or 'yourname=string'. **$(size)** is set to the size in bytes of this message. **$(reply-to)** is set to the Reply-To: field (or the From: field if the former is missing) and so can be used for automatic replies. **$(info)** is the info field from the internal mail header and is probably only of interest to the system administrators.

## *Example*

Here is a rough idea of what a *.maildelivery* file looks like:

```
# lines starting with a ``#'' are ignored.
# as are blank lines
# file mail with mmdf2 in the "To:" line into file mmdf2.log
To    mmdf2    file    A    mmdf2.log
# Messages from mmdf pipe to the program err-message-archive
From    mmdf    pipe    A    err-message-archive
# Anything with the "Sender:" address "uk-mmdf-workers"
# file in mmdf2.log if not filed already
Sender    uk-mmdf-workers    file    ?    mmdf2.log
# "To:" unix - put in file unix-news
To    Unix    >    A    unix-news
# if the address is jpo=mmdf - pipe into mmdf-redist
Addr    jpo=mmdf    |    A    mmdf-redist
# if the address is jpo=ack - send an acknowledgement copy back
Addr    jpo=ack    |    R    resend  -r  $(reply-to)
# anything from steve - destroy!
from    steve    destroy    A    -
# anything not matched yet - put into mailbox
default    -    >    ?    mailbox
# always run rcvalert
*    -    |    R    rcvalert
```

## *Files*

*$HOME/.maildelivery* normal location

## *See also*

**mmdftailor**(F), **rcvalert**(C), **rcvfile**(C), **rcvprint**(C), **rcvtrip**(C)

## *Credit*

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

# mmdftailor

provide run-time tailoring for the MMDF mail router

## *Description*

The MMDF mail router reads site-dependent information from the ASCII file */usr/mmdf/mmdftailor* each time it starts up.

Keywords in the tailor file are not case-sensitive; however, case is important for filenames and similar values. Use quotation marks to delimit strings to prevent them from being parsed into separate words accidentally.

The following alphabetical list describes most of the information you can set in the *mmdftailor* file. For information about additional channel-specific settings, refer to the documentation about the particular channel.

**ALIAS**
defines an alias table. The following parameters can be used:

**table** specifies the name of the table to be associated with this alias entry

**trusted** allows the entries in the table to cause delivery to files and pipes

**nobypass** does not allow the ***~address*** alias bypass mechanism to work on this file

Here is an example:

```
ALIAS table=sysaliases, trusted, nobypass
```

**AUTHLOG**
controls authorization information. See **MCHANLOG** and **MLOGDIR**.

**AUTHREQUEST**
is the address to which users should mail if they have questions about why a message was not authorized for delivery. It is also used as the sender of authorization warning messages. It is not used if authorization is not enabled on some channel. See the **auth** parameter under **MCHN**.

**MADDID**
controls whether **submit** adds `Message-ID:` header lines if they are missing from messages. It takes a number as an argument. If the number is 0, no action is taken. If the number is non-zero, then **submit** adds `Message-ID:` header lines if they are missing from messages.

**MADDRQ**
is the address files directory. If it is not a full pathname, it is taken relative to **MQUEDIR**.

**MCHANLOG**
controls MMDF logging, except for authorization information and information produced by **deliver** and **submit**. See also **MMSGLOG**, **AUTHLOG**, and **MLOGDIR**.

Logging files and levels can also be specified in the channel descriptions. The logging file, if specified there, overrides the **MCHANLOG** definition. The logging level for the channel is set to the maximum of the **MCHANLOG** level and the channel description's level. The **MCHANLOG** level can therefore be used to increase logging on all channels at once.

Here is an example:

```
MCHANLOG        /tmp/mmdfchan.log, level=FST, size=40,
                stat=SOME
```

An argument without an equal sign is taken as the name of the log. Logging levels are:

| | |
|---|---|
| **FAT** | logs fatal errors only |
| **TMP** | logs temporary errors and fatal errors |
| **GEN** | saves the generally interesting diagnostics |
| **BST** | shows some basic statistics |
| **FST** | gives full statistics |
| **PTR** | shows a program trace listing of what is happening |
| **BTR** | shows more detailed tracing |
| **FTR** | saves every possible diagnostic |

The **size** parameter is the number of 25-block units you will allow your log file to grow to. When a log file reaches that size, that logging either stops or cycles around overwriting old data (see **cycle**).

The **stat** parameter sets up various status flags for logging:

**close** closes the log after each entry; this allows other processes to write to it as well

**wait** if the log is busy, waits a while for it to free

**cycle** if the log is at the maximum length specified with the **size** parameter, then cycles to the beginning

**some** sets the values **close** and **wait** (the most common setting)

**timed** opens the log and, after the timeout period (for example, 5 minutes), closes the log and reopens it; this option overrides all other options (used to reduce the overhead of re-opening the log for every entry while still retaining the ability to move the log file out from under a running process and have the process begin logging in the new log file soon thereafter)

Tailoring of the log files is generally performed at the end of the tailor file to prevent logging the tailoring action itself, thereby needlessly filling the log files when higher tracing levels are enabled. If you have bugs in the tailoring, you can move the log-file tailoring closer to the top of the tailor file.

**MCHN**
defines a channel. The following parameters can be used:

**name** the name of the channel

**show** a descriptive name used by certain programs as a display line to explain the function of the channel

**que** the queue subdirectory of */usr/spool/mmdf/lock/home* in which to queue messages for this channel; MMDF prefixes the name with *q.* to form the subdirectory name.

**tbl** the abbreviated name (from **MTBL**) for the associated table that lists the hosts that are accessible on this channel. If the specified table has not been previously defined, it will be defined with the same **name**, **file**, and **show** parameters as for this channel (do not define two channels that process the same queue, but use different tables because it will cause queue structure problems).

**pgm** the channel program (in */usr/mmdf/chans*) to invoke for this channel. This program takes mail from **deliver**(ADM) and carries it to its destination on the local machine or across the network to a remote machine.

**mod** the delivery mode for the channel; if several values are selected, they are cumulative:

- **reg** regular mode (the default). This mode queues mail, but does not send it; you must run **deliver** (manually, with **cron**(C), or as a background program) to actually send mail through the regulated channel.
- **host** same as **reg**, but specifies that **deliver** sort by host after sorting by channel, which allows as many mail messages as possible to get sent to a particular host before the connection is broken.
- **bak** channel can be invoked only by the background deliver daemon
- **psv** channel is passive; it is a pickup-type channel (for example, pobox)

**imm** channel can be invoked immediately; no need to batch up mail

**pick** channel can pick up mail from the remote host

**send** channel can send mail to the remote host

**ap** the type of address parsing to use for reformatting headers on messages going out on this channel; if several values are selected, they are cumulative:

**same** does not reformat headers **822** converts to RFC822-style source routes (for example, @A:B@C) **733** converts to RFC733-style source routes (for example, B%C@A)

**nodots** selects output of leftmost part of domain names (for example, A in A.B.C) for sites that cannot handle domains (usually used in conjunction with the **known=** parameter to hide domain names behind a smart relay)

**lname** a name overriding the default **MLNAME** value for this channel (used when the channel should have non-standard values for the local domain)

**ldomain** a name overriding the default **MLDOMAIN** value for this channel

**host** the name of the host that is being contacted by this channel, usually used in the phone and pobox channels, or the name of the relay host when all mail to hosts in this channel's table gets relayed to one host (this is required on the *badusers* and *badhosts* pseudo-channels; it must be set to the local host for the list channel)

**poll** the frequency of polling the remote machine (0 disables polling, -1 requests polling to be done every time the channel is started up, any other value is the number of 15-minute intervals to wait between polls); if any mail is waiting to be sent, this value is ignored because a connection is always attempted

**insrc** a table of hosts controlling message flow

**outsrc** see **insrc**

**indest** see **insrc**

**outdest** see **insrc**

**known** a table of hosts that are known on this channel; be sure that the table contains your own fully specified host name

**confstr** a channel-specific configuration string. See the individual manual pages for the channel for more information.

**auth** specifies the authorization tests that are made on this channel:

- **free** default, no checking takes place
- **inlog** log information for incoming messages
- **outlog** log information for outgoing messages
- **inwarn** warn sender of incoming message if authorization is inadequate (the message still goes through)
- **outwarn** as **inwarn**, but for outgoing messages
- **inblock** reject incoming messages that have inadequate authorization
- **outblock** as **inblock**, but for outgoing messages
- **hau** host and user authorizations are required
- **dho** (direct host only) when applying host controls, the message must be associated with a user local to that host (that is, no source routes)

**ttl** (time-to-live) specifies the number of minutes for which retries to a host are blocked when **deliver** detects a connection failure to that host; this value can be overridden on the **deliver** command line (default is 2 hours)

**log** the name of the channel log file to be used instead of the default **MCHANLOG**

**level** the logging level for this channel (see also **MCHANLOG**)

Here is a simple example:

```
MCHN    name=local, que=local, tbl=local,
        show="Local Delivery", pgm=local,
        poll=0, mod=imm, ap=822, level=BST
```

If the first argument does not have an equal sign, the values of the **name**, **que**, **tbl**, **pgm**, and **show** parameters take on this value.

**MCHNDIR**
is where the channel programs are to be found.

**MCMDDIR**
is the default commands directory where the various MMDF commands are located. Any command that does not have a full pathname is taken relative to this directory.

**MDBM**
tells MMDF where to find the database file containing the associative store. DBM-style databases get their speed and flexibility by performing dynamic hashing on an associative store. This can get quite large. By default, the file is located in the **MTBLDIR** directory, but it might need to be relocated due to its size.

**MDFLCHAN**
sets the default channel to something other than local.

**MDLV**
is the name of the file used for tailoring the delivery for each user.

**MDLVRDIR**
is the directory in which to deliver mail. If this is null, then the user's home directory is used. See also **MMBXNAME** and **MMBXPROT**.

**MDMN**
defines a domain. The following parameters can be used:

**name** an abbreviated name for the domain

**show** a display line, which is used for formatting purposes to explain what the domain is all about

**dmn** the full name (x.y.z...) of this domain

**table** the associated table entry of known sites in this domain; if the specified table has not been previously defined, it will be defined with the same **name**, **file**, and **show** parameters as for this domain

Here is an example:

```
MDMN    name="Root", dmn="", show="Root Domain",
        table=rootdomain
```

If the first argument does not have an equal sign, the values of the **name**, **dmn**, and **show** parameters take on this value. If no **table** parameter is specified, it defaults to the value of the **name** parameter.

**MFAILTIME**
is the time a message can remain in a queue before a failed-mail message is sent to the sender and the message is purged from the queue. See also **MWARNTIME**.

**MLCKDIR**
is the directory where the locking of files takes place: this is dependent on what style of locking you are doing.

**MLCKTYPE**
specifies the locking protocol for MMDF to use when locking mailboxes. Use one or more of the following keywords with **MLCKTYPE**:

| Keyword | Lock File |
|---|---|
| **advisory** | System V **fcntl**( ) kernel file locking |
| **v7** | Version 7 and System V Release 3, and earlier file locking |
| **xenix** | XENIX file locking |
| **all** | all known locking protocols |

If you specify more than one locking keyword, all locks must be successful before MMDF considers the mailbox locked. Here is an example **MLCKTYPE** setting:

```
MLCKTYPE advisory, xenix
```

**MLDOMAIN**
gives your full local domain (this, combined with the **MLNAME**, and possibly the **MLOCMACHINE**, gives the full network address).

**MLISTSIZE**
specifies the maximum number of addresses in a message before it is considered to have a "big" list. If there are more than the maximum number of addresses, then MMDF does not send a warning message for waiting mail and only returns a "citation" for failed mail. A citation consists of the entire header plus the first few lines of the message body.

**MLNAME**
is the name of your machine or site as you wish it to be known throughout the network, which can be a generic host name used to hide a number of local hosts. If it is a generic host name, internal hosts are differentiated by **MLOCMACHINE**. See also **MLDOMAIN**.

**MLOCMACHINE**
is the local name of the machine. It is used by a site that has several machines, but wants the machines themselves to appear as one address. See also **MLNAME** and **MLDOMAIN**.

**MLOGDIR**
is the default directory in which the log files are kept. See also **MMSGLOG**, **NAUTHLOG**, and **MCHANLOG**.

**MLOGIN**
is the user who owns the MMDF transport system.

**MMAXHOPS**
specifies the maximum number of `Received:` or `Via:` lines in a message before it is considered to be looping and is rejected.

**MMAXSORT**
controls sorting of messages based on the number of messages in the queue. If the number of messages in the queue is less then **MMAXSORT**, the messages are sorted by arrival time and are dispatched in that order; otherwise, a message is dispatched as it is found during the directory search.

**MMBXNAME**
is the name of the mailbox. If this is null, then the user's login name is used. See also **MDLVRDIR** and **MMBXPROT**.

**MMBXPREF**
is a string written before the message is written into the mailbox. It is usually set to a sequence of ⟨Ctrl⟩**A** characters. The default **MMBXPREF** value looks like this:

```
MMBXPREF "\001\001\001\001\n"
```

See also **MMBXSUFF**.

The values of **MMBXPREF** and **MMBXSUFF** should consist of non-printable characters only and must end in a newline.

**MMBXPROT**
gives the protection mode in octal for the user's mailbox. See also **MDLVRDIR** and **MMBXNAME**.

**MMBXSUFF**
is a string written after the message is written into the mailbox. It is usually set to a sequence of ⟨Ctrl⟩**A** characters. The default **MMBXSUFF** value looks like this:

```
MMBXSUFF "\001\001\001\001\n"
```

See also **MMBXPREF**.

**MMSGLOG**
controls the logging information produced by **deliver** and **submit**. See also **MCHANLOG**, **AUTHLOG**, and **MLOGDIR**.

**MMSGQ**
is the directory for the files of message text. If it is not a full pathname, it is taken relative to **MQUEDIR**.

**MPHSDIR**
is the directory in which the timestamps for the channels are made, showing what phase of activity they are in.

**MQUEDIR**
is the parent directory for the various queues and address directories.

**MQUEPROT**
gives the protection mode in octal that the parent of the **MQUEDIR** directory should have.

**MSIG**
is the signature that MMDF uses when notifying senders of mail delivery problems.

**MSLEEP**
is the length of time in seconds that the background delivery daemon sleeps between queue scans.

**MSUPPORT**
is the address to which to send mail that MMDF cannot cope with (that is, that MMDF cannot deliver or return to its sender).

**MTBL**
defines an alias, domain, or channel table. The following parameters can be used:

**name** a short name by which the table can be referred to later in the file

**file** the file from which the contents of the table are built

**show** a display line, which is used for reporting purposes to explain what the table is all about

**flags** indicates additional properties about the table being defined. Use this to specify the source type, nameserver lookup parameters, and control of partial lookups table options.

The following three source types that define the table:

**file** comes from a sequentially read file (default).

**dbm** is in the MMDF hashed database built with **dbmbuild**.

**ns** the table data is obtained by making queries on a nameserver.

**domain** specifies to look up the given address in the domain specified by **dmn=** parameter of the domain definition.

**channel** specifies to look up the given fully-qualified domain name to determine the address(es) to use in delivering via SMTP.

**alias** specifies to look up the given alias name in alias tables.

**abort** specifies that if MMDF encounters a problem when searching an **ns**-type domain table, MMDF does not search any other domain tables (because the **ns**-type domain table is the most reliable).

**route** specifies to search for successive subdomains of the domain if no exact match exists.

**partial** specifies to search for the domain in other domain tables; this allows users to omit the full domain specification when referring to local machines.

Note that MMDF treats **flags=file** and **flags=dbm** the same. In the case of an **ns**-type table, the **flags** field specifies the type of nameserver lookup (either **domain**, **channel**, or **alias**).

A typical example might be:

```
MTBL   name=aliases, file=aliases,
       show="User & list aliases"
```

If the first argument does not have an equal sign, the values of the other parameters take on this value. The following example sets the **name**, **file**, and **show** parameters to the string "aliases", then resets the **show** parameter to the string "Alias table".

```
MTBL aliases, show="Alias table"
```

**MTBLDIR**
is the default directory for the table files.

**MTEMPT**
is the temporary files directory. If it is not a full pathname, it is taken relative to **MQUEDIR**.

**MWARNTIME**
specifies the time in hours that a message can remain in a queue before a warning message about delayed delivery is sent to the sender. See also **MFAILTIME**.

**UUname**
defines the UUCP sitename (short form, not full path) and is used only by the UUCP channel. See also **MLNAME**.

## *See also*

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

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

## *Value added*

**mmdftailor** 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.

# queue

MMDF queue files for storing mail in transit

## *Description*

MMDF stores mail in */usr/spool/mmdf/lock/home*, an isolated part of the filesystem, so that only authorized software may access the mail. Mail is stored under the directory sub-tree.

It must specify a path with at least two sub-directories. The next-to-bottom one is used as a "locking" directory and the bottom one is the "home". For full protection, only authorized software can move through the locking directory. Hence, it is owned by MMDF and accessible only to it.

### *Queue entries*

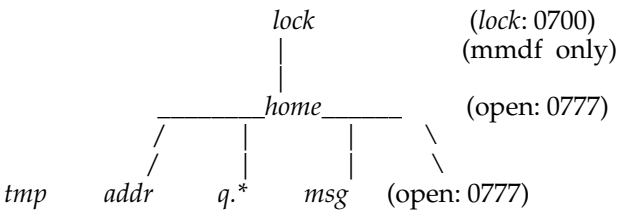
When mail is queued by **submit**, it is actually stored as two files. One contains the actual message text and the other contains some control information and the list of addressees.

The text file is stored in the *msg* directory. The other file is stored in the *addr* directory and is linked into one or more queue directories. The queue directories are selected based on the channels on which this message will be delivered. Each output channel typically has its own queue directory.

Another directory below *home* is a temporary area called *tmp*. It holds temporary address-lists as they are being built. Queuing of a message is completed by linking this address file into *addr* and the queue directories. The *msg* directory contains files with message text. *Addr* and *msg* files are synchronized by giving them the same unique name, which MMDF occasionally calls the message "name". The message name is derived by use of **mktemp**(S), using *msg* as the base directory. The files created in *addr* must have open read-write access; the ones in *msg* must have open read access.

When **submit** runs, it changes into *home* for its working directory. It then does a **setuid** to run as the caller. This is necessary to permit **submit** to access the caller's address-list files (specified as "**<** file"). **Deliver** changes into the queue directory to minimize the time spent searching for messages to deliver.

The following depicts the directory organization:




addresses ==> moved and linked message text
built here ==> into here into here put here

entries: 0666 0666 0644

## Queue file formats

The *msg* portion of a queued message simply contains the message, which must conform to the Arpanet standard syntax, specified in RFC822. It is expected that the format of the message contents file eventually will be more structured, permitting storage of multi-media mail.

The following specifies the syntax of the *addr* (and queue directory) address-list portion of the queued message:

## Address file contents

| | |
|---|---|
| file ::= | creation late flags '\ n' [rtrn-addr] '\ n' *(adr_queue '\ n') |
| creation ::= | {long integer decimal representation of the time the message was created} |
| late ::= | ADR_MAIL / ADR_DONE {from adr_queue.h} |
| flags ::= | {decimal representation of 16-bits of flags} |
| rtrn-addr ::= | {rfc822 return address} |
| adr_queue ::= | temp_ok mode queue host local {conforms to structure specified in adr_queue.h} |
| temp_ok ::= | {temporary flag indicating whether this address has been verified by the receiving host: "yes" is " + "; "not yet" is " - "} |
| mode ::= | {send to mailbox(m), tty(t), both(b), either(e), or processing completed(*)} |
| queue ::= | {name of the queue into which this message is linked for this address} |

host ::= {official name (and domain) of recipient host}

local ::= {local address on receiving host}

## *Address file description*

An address queue file contains a creation time-stamp, an indication if the sender has been notified of delayed delivery, some flags, an optional return-mail address, and an address list. Several <flags> are currently in use. 0004 indicates whether late warnings should be sent to the return-mail address if the entire address list has not been processed within the number of hours specified by "warntime". 0002 indicates whether mail should be returned to the sender if it hasn't been completely processed within the number of hours specified by "failtime". 0001 indicates whether warning and failure messages are to contain only a citation of the message. An "*" value, for the "late" flag, indicates that a warning notice has been sent.

The creation date is coded as a long ASCII decimal string, terminated by the "late" flag. This has to be inserted into the file because doesn't maintain it. The date is used to sort the queue so that mail is delivered in the order submitted.

The return address is a line of text and may be any address acceptable to **submit.**

Each address entry is on a separate line, and conforms to the adr_struct format, defined in *adr_queue.h.* It contains several fields separated by spaces or commas. Fields containing spaces or commas must be enclosed in double quotes.

The temp_ok flag indicates whether the address has been accepted during an "address verification" dialog with the receiving host. When the message text is successfully accepted by the receiving host, then this flag no longer applies and the mode is set to ADR_DONE ("*"). Before final acceptance of message text, the mode flag indicates whether the mail is for a mailbox, terminal, both, or either. (Currently only mailbox delivery, ADR_MAIL, is used.)

The queue name is the name of the sub-queue in which the message is queued for this address. Each addressee's host may be on a separate queue or some hosts may share the same queue. When all addressees in the same queue have been delivered, the address file is removed from that queue directory. When all queues have been processed, the address file (in both *addr* and the queue directory) and the text file (in *msg*) are removed.

The host and local strings are used by the channel program. The host determines the type of connection the channel program makes. The local string is passed to the host; it should be something meaningful to that host. The local string must not contain newline or null and it must be a valid local address per RFC822.

## *See also*

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

## *Credit*

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

# tables

MMDF name tables for aliases, domains, and hosts

## *Description*

All of the MMDF name tables are encoded into a database which is built on top of the **dbm**(S) package. A number of tables are associated with MMDF, the exact set being specified by the tailor file, */usr/mmdf/mmdftailor*. Name tables all have the same format. Functionally, they permit a simple key/value pairing. The syntax for tables is specified here:

| | |
|---|---|
| entries ::= | entries entry |
| entry ::= | comment / real-entry |
| comment ::= | '#' value eol |
| real-entry ::= | name separator value eol |
| name ::= | {string of chars not containing a <separator>} |
| separator ::= | {':' and whitespace} |
| value ::= | {string of chars not containing an <eol>} |
| eol ::= | {newline, nullch, DEL} |
| where: | |
| name is | a key |
| value is | any relevant text. |

### *Hosts and domains*

Two basic types of table are host and domain tables. This section gives a brief discussion of these concepts in terms of the MMDF system. The domain namespace is treated as a logical global hierarchy, according to the model of RFC 819, with subdomains separated by "." (for example, ISI.USC.ARPA is a three level hierarchy with ARPA at the top level). A host is a computer associated with a channel which may be directly connected or reached through a relay associated with the channel. The distinction between hosts as physical entities, and domains as logical entities should be noted. All hosts known to an MMDF system must have unique names. For this reason, the convention of labeling hosts by an associated domain name is adopted in many cases. This is a useful method to guarantee unique names, but is not required.

The domain and host table structures are devised with three basic aims in mind:

1. To map a string into a fully expanded domain name.
2. To map this domain into a source route starting with a host.
3. To obtain the transport address associated with the host.

## *Domain tables*

Domains are split in a two-level manner, with the top part of the tree specified in the tailor file and the lower parts of the tree in tables. The two-level structure is intended as a balance between generality and efficiency. The order of searching is also specified in the tailor file. The structure of a domain table is to have **name** as the part of the domain not in the tailor file. Thus for ISI.USC.ARPA there might be a domain ARPA with name=isi.usc or domain USC.ARPA with name=isi. The structure of value is:

value ::= *(domain dm_separator) host

The possible values of **dm_separator** are given in **tai**(S), although typically " , " or ' ' would be used. This value is essentially a source route to be traversed from right to left. Consider an example table for the domain ARPA:

```
# Sample ARPA domain table
isi.usc:a.isi.usc.arpa
b.isi.usc:b.isi.usc.arpa
foobar.isi.usc:b.isi.usc.arpa
graphics.isi.usc:graphics.isi.usc.arpa z.mit.arpa
```

Thus, if the "isi.usc.arpa" is analyzed, domain table ARPA will be selected, and host "a.isi.usc.arpa" associated with domain "isi.usc.arpa." If only "isi.usc" were given, the domain tables would be searched in order, and if the ARPA table were the first one to give a match, then the same result would be reached. If "foobar.isi.usc" is given, it would be mapped to host "b.isi.usc.arpa" and (official) domain "b.isi.usc.arpa." If "graphics.isi.usc.arpa" is given, a source route to domain "graphics.isi.usc.arpa" through HOST "z.mit.arpa" will be identified. If "xy.isi.usc.arpa" (or "xy.isi.usc") is given, then it will not be found. However, a subdomain will be stripped from the left and the search repeated. Thus domain "xy.isi.usc.arpa" will be identified as reached by a source route through host "a.isi.usc.arpa."

As specified earlier, the order of searching is also specified in the tailor file. For example, a host in domain UCL-CS.AC.UK, might have a search order UCL-CS.AC.UK, AC.UK, UK, SWEDEN, ARPA, "". Thus, if there were a domain isi.usc.ac.uk, it would be the preferred mapping for isi.usc over isi.usc.arpa. The last domain searched is null. This could be used to contain random fully qualified domains or to identify gateways to other domains. An example file is:

```
#  Sample Top level domain table
#  Odd host
basservax.australia:basservax.australia scunthorpe.ac.uk
# UUCP Gateway
uucp:seismo.arpa
# Mailnet Gateway  (-> multics -> educom ->mailnet)
mailnet:educom.mailnet mit-multics.arpa
```

To specify the top domain in the tailor file, the name and dmn parameters of the domain should be set to "".

## *Host tables*

For every host associated with the channel, there will be one or more entries. In each case, the key is the name identified from the domain tables. A host may have multiple entries if it has more than one transport address which the channel might utilize.

When a channel just sends all its mail to a relaying site, the address portion (value) of the entry is not needed, directly, during the transmission process. Hence, it need not be accurate. However, it is still used to logically collect together host names, that is, all table entries with the same value are regarded as being aliases for the same host.

## *P.O. box channels*

POBox channels, for passive, telephone-based exchange, operate in two modes. In the first mode, a single login is authorized to pickup all mail for the channel. In this case, the host-table addresses are only used for the "collecting" function. For the second mode, different logins share the channel and are to receive only some of the mail queued for the channel. In this case, the login is treated as an "address", and the table entries should have the value fields contain the name of the login authorized to pickup mail for that "host".

## *Access control tables*

Channels also have access control tables associated with them, to determine whether a message is allowed to use a given route. Each channel has four (optional) tables that determine the access controls used: insrc, outsrc, indest, and outdest.

## *Reformatting tables*

There may also be a “known hosts” table associated with each channel. This is exactly the same format as a host table. If a message is being reformatted, and if an address does not have its host in this list, then it will be modified to appear as a percent route (RFC733 or JNT Mail route) address, with the local domain as the root.

## *Local aliases*

The password file specifies the mailing names are login names of all local recipients. Since this is a rather restricted name space, and since it is useful to have some other kinds of locally-known names, there is a second file used to specify "aliases". The location of the aliases file is specified in the tailor file.

An alias entry may be used for one of five functions:

1. True aliasing, where the key value maps to a local user's login name, for example "dave:dcrocker;"

2. Forwarding, where the key value maps to a foreign address, such as "dcrocker:dcrocker@udel;" and

3. Address lists, where the key value maps to a set of addresses, such as "mother:cotton,dcrocker,farber."

4. Redirection of a message to a file: for example, "foobar:dpk/foobar" would cause user and group ids to be set to dpk and the text of the message to be appended to the file "foobar" in dpk's default login directory. Similarly, "foobar:dpk//tmp/foobar" would do the same for file */tmp/foobar*.

5. Redirection of a message to a pipe. For example, "news-inject:news|/usr/lib/news/uurec" would cause a message to be passed into a UNIX pipe (see **pipe**(S)) with userid and groupid set to **news**.

As a convenience, the value-part of an entry may specify a file name, so that the **actual** value is taken from the file. There are two possible notations for this:

1. By having left-angle bracket ("<") precede the value specification. For example: "mother: < /etc/mmdf/mother_list@udel-relay.arpa."

2. By using a data type with value "include." For example: "mother: :include: /etc/mmdf/mother@udel-relay.arpa"

In both cases, the @HOST (not a domain) is optional. If specified, it should be the local host.

Recursive specification is permitted. For example, "crocker" may map to "dcrocker" and "dcrocker" may map to "dcrocker at udel," so that both "crocker" and "dcrocker" are locally-known names, but mail sent to either of them will be forwarded to "dcrocker@udel."

In practice, it is useful to organize alias files into the following order:

List aliases
: which contain a value referring to a later address list. This constitutes a one-to-one mapping of a key to a value, where the value points into the "Lists" group.

Lists
: which contain values referring to multiple addresses: this constitutes a one-to-many mapping, where some of the addresses may refer to other entries (address lists) in the Lists group, as well as other entries (individual addresses) later in the table.

Mailbox aliases
: which contain values referring to single addresses. These constitute one-to-one mappings, where the value refers to an entry in the password file or to an entry in the "Forwarding aliases" group.

Forwarding aliases
: which contain values referring to single addresses on other machines. These, also, are one-to-one mappings, where the value always refers to an off-machine address.

By organizing the file in this manner, only the "Lists" portion requires a topological sort. Since the other three sections will never point to entries within their section, they may be sorted more conveniently, such as alphabetically. Such a structure also tends to make changes easy. In particular, the handling of forwarding is easy, since **all** references to a user will get intercepted, at the end of the table.

### *Mail-ID tables*

The Mail-ID tables are used only if the Mail-IDs feature is enabled. This can be done in the tailoring file, by defining MMAILID to be 1. Mail-IDs are used to disassociate mail addresses from login names. There are two tables that are used to map Mail-IDs to users' login names and login ids to Mail-IDs. The "users" file is used to map users (login ids) to Mail-IDs, and the "mailids" file is used to map Mail-IDs to users. The names of these files can be overridden, but it is not recommended. Each file has lines with two entries per line (user and Mail-ID, or Mail-ID and user).

A user can have more than one entry in the Mail-IDs file, but should have only one entry in the users file. This does not prevent them from sending mail with any of their Mail-IDs. The users file is just a source of default Mail-IDs.

## *Value added*

**tables** 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.