

NAME

update-mime – create or update MIME information

SYNOPSIS

update-mime [no parameters]

DESCRIPTION

update-mime updates the `/etc/mailcap` file to reflect mime information changed by a Debian package during installation or removal.

OPTIONS

--local Generate files in the current user's home directory instead of the `/etc` directory. This allows users to create a custom ordering configuration and get a complete `%.mailcap` file out of it. In this local mode, the order overriding file (see below) will be looked for in the `%.mailcap.order` file.

OVERRIDING ORDER

The order of entries in the `/etc/mailcap` file can be altered by editing the `/etc/mailcap.order` file. Please see the **mailcap.order(5)** man page for more information.

CREATING ENTRIES

To create entries in the mailcap file, packages need to create a file in the `/usr/lib/mime/packages` directory. In this file goes the verbatim desired mailcap entries. In addition to the standard mailcap options (described below) is a new *priority* option. Specifying this will provide for simple ranking of programs within a given mime type. An animation viewer, for example, may be able to display a static picture, but probably wouldn't be the best choice and so would give an option like "priority=2". Priorities range from 0 to 9, with 0 being the lowest and 9 being the highest. If the *priority* option is omitted, a value of 5 is used.

The following are standard options that can be specified in the mailcap entry. Options are separated by semicolons (;) but must all be on the same line. Each line should look like:

```
mime/type; viewer; option; another=val; etc; priority=5
```

Mime types of the form "class/*" and even "*/*" are now acceptable (they were previously disallowed). When using "class/*", it is probably a good idea to add a "priority=[1-4]" option so specific rules using the default priority will get chosen first. If using "*/*", though, you probably want to add a "priority=0" option to make that rule a "last resort".

Commands**<program-string>**

Specifies the program to run to view a file of the given content-type. **This option setting cannot be omitted.** An implicit "view=" can be considered before it. When writing an entry that has no viewer, use a value of *false* in this space.

compose=<program-string>

The "compose" command may be used to specify a program that can be used to compose a new body or body part in the given format. Its intended use is to support mail composing agents that support the composition of multiple types of mail using external composing agents. The result of the composing program may be data that is not yet suitable for mail transport -- that is, a Content-Transfer-Encoding may need to be applied to the data.

composetyped=<program-string>

The "composetyped" command is similar to "compose", but is to be used when the composing program needs to specify the Content-type header field to be applied to the composed data. The "compose" option is simpler, and is preferred for use with existing (non-mail-oriented) programs for composing data in a given format. The "composetyped" option is necessary when the Content-type information must include auxiliary parameters, and the composition program must then know

enough about mail formats to produce output that includes the mail type information.

edit=<program-string>

The "edit" command may be used to specify a program that can be used to edit a body or body part in the given format. In many cases, it may be identical in content to the "compose" command.

print=<program-string>

The "print" command may be used to specify a program that can be used to print a message or body part in the given format.

Modifiers

These options are modifiers to all the commands specified on the command line.

test=<conditional>

The "test" option may be used to test some external condition (e.g., the machine architecture, or the window system in use) to determine whether or not the mailcap line applies. It specifies a program to be run to test some condition. If the test fails, a subsequent mailcap entry will be sought. Multiple test options are not permitted -- since a test can call a program, it can already be arbitrarily complex.

Note: When testing for X by looking at the *DISPLAY* environment variable, please use one of:

```
test=test -z "$DISPLAY"   (no X)
or test=test -n "$DISPLAY" (have X)
```

Many programs recognize these strings and optimize for them.

needsterminal

The "needsterminal" option, if given, indicates that the commands must be run on an interactive terminal. This is needed to inform window-oriented user agents that an interactive terminal is needed. (The decision is not left exclusively to the command because in some circumstances it may not be possible for such programs to tell whether or not they are on interactive terminals.) The needsterminal command applies to the view, compose and edit commands, if they exist. Note that this is NOT a test -- it is a requirement for the environment in which the program will be executed, and will typically cause the creation of a terminal window when not executed on either a real terminal or a terminal window.

copiousoutput

The "copiousoutput" option, if given, indicates that the output from the view-command will be an extended stream of output and is to be interpreted as advice to the UA (User Agent mail-reading program) that the output should be either paged or made scrollable. Note that it is probably a mistake if needsterminal and copiousoutput are both specified.

Content-Type Info

These options provide additional information about the given content-type.

description=<string>

The "description" option simply provides a textual description that describes the type of data, to be used optionally by mail readers that wish to describe the data before offering to display it.

textualnewlines

The "textualnewlines" option, if given, indicates that this type of data is line-oriented and that, if encoded in a binary format, all newlines should be converted to canonical form (CRLF) before encoding, and will be in that form after decoding. In general, this is needed only if there is line-oriented data of some type other than text/* or non-line-oriented data that is a subtype of text.

x11-bitmap=<pathname>

The "x11-bitmap" option names a file, in X11 bitmap (x11) format, which points to an appropriate icon to be used to visually denote the presence of this kind of data.

nametemplate=<string>

The "nametemplate" option gives a file name format, in which %s will be replaced by a short unique string to give the name of the temporary file to be passed to the viewing command. This is only expected to be relevant in environments where filename extensions are meaningful, e.g., one could specify that a GIF file being passed to a gif viewer should have a name ending in ".gif" by using "nametemplate=%s.gif".

DEPENDENCIES

Packages that wish to provide MIME access to themselves should **not** depend on, recommend, or suggest **mime-support**, as the file they create in */usr/lib/mime/packages* will cause **update-mime** to be automatically run via a Dpkg trigger.

DESKTOP ENTRIES

In addition to the abovementioned mechanism **update-mime** also parses desktop entries in */usr/share/applications/* to generate mailcap entries. These entries are given a lower priority than those in */usr/lib/mime/packages*.

SEE ALSO

mailcap.order(5), **deb-triggers(1)**, RFC-2046, RFC-1524

AUTHOR

update-mime was written by Brian White <bcwhite@pobox.com>

COPYRIGHT

update-mime is in the public domain (the only true "free").