Network Working Group
Internet Draft

N. Borenstein
Bellcore
June 1991

A User Agent Configuration Mechanism

For Multimedia Mail Format Information

Abstract

This memo suggests a file format to be used to inform multiple mail reading user agent programs about the locally-installed facilities for handling mail in various formats. The mechanism is explicitly designed to work with mail systems based on either X.400 or Internet mail (as defined by RFC's 821, 822, 934, 1049, 1113, and an a concurrently-developed memo defining "multipart" mail and encodings for 8-bit data). The format and mechanism are proposed in a manner that is generally operating-system independent. However, certain implementation details will inevitably reflect operating system differences, some of which will have to be handled in a uniform manner for each operating system. This memo makes such situations explicit, and, in an appendix, suggests a standard behavior under the UNIX operating system.

Status of This Memo

This memo describes an Experimental Protocol for the Internet community, and requests discussion and suggestions for improvements. This memo does not specify an Internet standard. Distribution of this memo is unlimited.

Introduction

The electronic mail world is in the midst of a transition from single-part text-only mail to multi-part, multi-media mail. In support of this transition, various extensions to RFC 821 and RFC 822 have been proposed and/or adopted. Various parties have demonstrated extremely high-functionality multimedia mail, but the problem of mail interchange between different user agents has been severe. In general, only text messages have been shared between user agents that were not explicitly designed to work together. This limitation is not compatible with a smooth transition to a multi-media mail world.

One approach that has been recently proposed by IFIP Working Group 6.5 and prototyped at Bellcore is to modify diverse sets of mail reading user agents so that, when they need to display mail of an unfamiliar (non-text) type, they consult an external file for information on how to display that file. That file might say, for example, that if the

content-type of a message is "foo" it can be displayed to the user via the "displayfoo" program.

This approach means that, with a one-time modification, a wide variety of mail reading programs can be given the ability to display a wide variety of types of message. Moreover, extending the set of media types supported at a site becomes a simple matter of installing a binary and adding a single line to a configuration file. Crucial to this scheme, however, is that all of the user agents agree on a common representation and source for the configuration file. This memo proposes such a common representation.

Location of Configuration Information

Each user agent must clearly obtain the configuration information from a common location, if the same information is to be used to configure all user agents. However, individual users should be able to override or augment a site's configuration. The configuration information should therefore be obtained from a designated set of locations. The overall configuration will be obtained through the virtual concatenation of several individual configuration files known as mailcap files.

The precise location of the mailcap files is operating-system dependent. A standard location for UNIX is specified in Appendix A.

Overall Format of a Mailcap File

Each mailcap file consists of a set of entries that describe the proper handling of one media type at the local site. For example, one line might tell how to display a message in Group III fax format. A mailcap file consists of a sequence of such individual entries, separated by newlines (according to the operating system's newline conventions). Blank lines and lines that start with the "#" character (ASCII 35) are considered comments, and are ignored. Long entries may be continued on multiple lines if each non-terminal line ends with a backslash character ('\', ASCII 92), in which case the multiple lines are to be treated as a single mailcap entry.

Thus the overall format of a mailcap file is given, in the modified BNF of RFC 822, as:

```
Mailcap-File = *Mailcap-Line

Mailcap-Line = Comment / Mailcap-Entry

Comment = NEWLINE  /  "#" *CHAR NEWLINE

NEWLINE = CRLF or NL, or as defined by OS convention
```

Format of a Mailcap Entry

Each mailcap entry consists of a number of fields, separated by semi-colons. The first three fields are required, and must occur in the specified order. The remaining fields are optional, and may appear in any order.

The first field is the content-type, which indicates the type of data this mailcap entry describes how to handle. It should correspond exactly (though case-insensitively) to an XXXX-Conformant "Content-type" header for Internet mail, or to a body-part name in X.400 mail. An optional second part of this field is the subtype specification, which is a forward slash ("/") followed by a subtype. If the subtype is specified as "*", it is intended to match all subtypes of the named content-type.

The second field, view-command, is a specification of how the message or body part can be viewed at the local site. Although the syntax of this field is fully specified , its semantics are necessarily somewhat operating system dependent. UNIX semantics are given in Appendix A.

The third field, referred to here as "notes," is an uninterpreted textual comment. It is suggested for use in documenting the evolution of a site’s mailcap file.

The optional fields, which may be given in any order, are as follows:

-- The "compose" flag may be used to specify a program that can be used to compose a new message 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. As with the view-command, the semantics of this flag are operating system dependent, with UNIX semantics specified in Appendix A.

-- The "print" flag may be used to specify a program that can be used to print a message or body part in the given format. As with the view-command, the semantics of this flag are operating system dependent, with UNIX semantics specified in Appendix A.

-- The "needsterminal" flag indicates that the view-command 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 view-command because in some circumstances it may not be possible for such programs to tell whether or not they are on interactive terminals.)

-- The "copiousoutput" flag 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.

In RFC 822 modified BNF, the following grammar describes a mailcap entry:

```
Mailcap-Entry = type ["/" wildsubtype] ; view-command
;
```

```
                              notes [";" 1#flag]

wildsubtype = subtype / "*"

view-command = mtext

notes = mtext

mtext = quoted-string / unquoted-no-semi

unquoted-no-semi = * <any CHAR except
                              ";", <"> and CRLF/NL>

flag :="needsterminal"    ; All these literals are to
       / "copiousoutput"          ; be interpreted as
       / "compose=" mtext         ; case-insensitive
       / "print=" mtext
```

Note that "quoted-string" is defined in RFC 822, and that "type" and "subtype" are defined in Internet Draft XXXX.

Appendix A: Implementation Details for UNIX

Although this memo fully specifies a syntax for "mailcap" files, the semantics of the mailcap file are of necessity operating-system dependent in four respects. In order to clarify the intent, and to promote a standard usage, this appendix proposes a UNIX semantics for these four cases. If a mailcap mechanism is implemented on non-UNIX systems, similar semantic decisions should be made and published.

Location of the Mailcap File(s)

For UNIX, a path search of mailcap files is specified. The default path search is specified as including at least the following:

$HOME/.mailcap:/etc/mailcap:/usr/etc/mailcap:/usr/local/etc/mailcap

However, this path may itself be overridden by a path specified by the MAILCAPS environment variable.

Semantics of the view-command

On a UNIX system, it will be a full shell command line, including path name for a program and its arguments. (Because of differences in shells and the implementation and behavior of the same shell from one system to another, it is recommended that the command line be a simple one that will be interpreted identically by all shells. If necessary, the command could itself run a shell script in a known language.) The body part will be passed to this command as standard input unless one or more instances of

"%s" appear in the view-command, in which case they will be replaced by the name of a file containing the body part, a file which may have to be created before the view-command program is executed. Additionally, any occurence of "%p" in the view-command will be replaced by the Content-type parameters (everything on the content-type header line following the first semicolon). Furthermore, any occurrence of "%t" will be replaced by the content-type subtype specification. (That is, if the content-type is "text/iso-646", then %t will be replaced by "iso-646".) A literal % character may be quoted as %%.

Semantics of the "compose" flag

On UNIX, the composing program is expected to produce a data stream for such a body part as its standard output.

By default, the composed message will be given a Content-type field that has no version number or resource reference, e.g.

Content-type: foobar

If this is unacceptable, the composing program can set the environment variable CONTENTTYPE to a string which will become the entire Content-type header field. Thus, the compose program might set $CONTENTTYPE to be "foobar; 42; math-lib" to indicate that the correct header to be applied is

Content-type: foobar; 42; math-lib

This, according to RFC 1049 and XXXX, specifies a content-type of "foobar", with a version string of "42" and a resource-reference of "math-lib".

Semantics of the "print" flag

On UNIX, the "print" program expects to receive the object on standard input unless one or more instances of "%s" are present. The "%s", "%p", and "%t" conventions of the view-command are also applied to the print command.

Appendix B: Sample Mailcap File

The following is an example of a mailcap file for UNIX that demonstrates most of the syntax above. It contains explanatory comments where necessary.

```
# Mailcap file for Bellcore lab 214.
#
# The next line handles Andrew format, if ezview is installed
x-be2; /usr/local/pkg/X11/andrew/bin/ezview %s; \
   nsb-installed 12/5/90; copiousoutput ; \
   print=/usr/local/pkg/X11/andrew/bin/ezprint %s
#
# Next, SPARC audio, uuencoded or via local file reference
x-sparc-audio-uu; uudecode < %s > /dev/audio ; nsb 1/8/91
x-sparc-audio-file; cat ‘cat %s‘ > /dev/audio  ; nsb 1/8/91
#
# Next, use the xview program to handle several image formats
x-xbm; xview %s; nsb 1/8/91
x-ppm-file; xview ‘cat %s‘; nsb 1/8/91
x-xview-uu; uudecode < %s | xview stdin; nsb 1/8/91
```

```
x-ppm-uu; uudecode < %s | xview stdin; nsb 1/8/91
#
# The MAGICMAIL interpreter uses curses, so needs a terminal
magicmail; /usr/local/bin/magicmail %s; nsb-installed 12/5/90 ; \
    needsterminal
```

Appendix C: A Note on Format Translation

It has been suggested that another function of a mailcap-like mechanism might be to specify the locally available tools for document format translation. For example, the file could designate a program for translating from format A to format B, another for translating from format B to format C, and finally a mechanism for displaying format C. Although this mechanism would be somewhat richer than the current mailcap file, and might conceivably also have utility at the message transport layer, it significantly complicates the processing effort necessary for a user agent that simply wants to display a message in format A. Using the current, simpler, mailcap scheme, a single line could tell such a user agent to display A-format mail using a pipeline of translators and the C-format viewer. This memo resists the temptation to complicate the necessary processing for a user agent to accomplish this task. Using the mailcap format defined here, it is only necessary to find the correct single line in a mailcap file, and to execute the command given in that line.

References

[RFC 822] Crocker, D., "Standard for the format of ARPA Internet text messages", RFC 822, Network Information Center, SRI International, August, 1982

[RFC 1049] Sirbu, M.A., "Content-type header field for Internet messages", RFC 1049, Network Information Center, SRI International, March, 1988

[XXXX] Borenstein, N., and N. Freed, "Mechanisms for Specifying and Describing the Format of Internet Message Bodies", Internet Draft, to be published simultaneously with this memo.

Author's Address

Nathaniel S. Borenstein
MRE 2A-274, Bellcore
445 South St.
Morristown, NJ 07962-1910
Email: nsb@bellcore.com
Phone: +1 201 829 4270
Fax: +1 201 829 7019