Network Working Group
Internet Draft: draft-faltstrom-macmime1-00
Expiration 4/94

Patrik Fältström
Dave Crocker
Erik E. Fair
October 15, 1993

MIME Encapsulation of Macintosh files - MacMIME

1. Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents at any timne. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress". Please check the `1idab-stracts.txt` listing contained in the internet-drafts Shadow Directories on `nic.ddn.mil`, `nnsc.nsf.net`, `nic.nordu.net`, `ftp.nisc.sri.com`, or `munnari.oz.au` to learn the current status of any Internet Draft.

2. Abstract

This memo describes the format to use when sending Apple Macintosh files via MIME [BORE93]. The format is compatible with existing mechanisms for distributing Macintosh files, while allowing non-Macintosh systems access to data in standardized formats.

3. Introduction

Files on the Macintosh consists of two parts, called forks:

Data fork: The actual data included in the file. The Data fork is typically the only meaningful part of a Macintosh file on a non-Macintosh computer system. For example, if a Macintosh user wants to send a file of data to a user on an IBM-PC, she would only send the Data fork.

Resource fork: Contains a collection of arbitrary attribute/value pairs, including program segments, icon bitmaps, and parametric values.

Additional information regarding Macintosh files is stored by the Finder has in a hidden file, called the "Desktop Database".

Because of the complications in storing different parts of a Macintosh file in a non-Macintosh filesystem that only handles consecutive data in one part, it is common to convert the Macintosh file into some other format before transferring it over the network.

The two styles of use are [APPL90]:

AppleSingle:	Apple's standard format for encoding Macintosh files as one byte stream.
AppleDouble:	Similar to AppleSingle except that the Data fork is separated from the other parts of the Macintosh File.

AppleDouble is the preferred format for a Macintosh file that is to be included in an Internet mail message, because it provides recipients with Macintosh computers the entire document, including Icons and other Macintosh specific information, while other users easily can extract the Data fork (the actual data) as it is separated from the AppleDouble encoding.

4. MIME format for Apple/Macintosh-specific file information

4a. APPLICATION/APPLEFILE

MIME type-name:	APPLICATION
MIME subtype name:	APPLEFILE
Required parameters:	none
Optional parameters:	NAME, which must only consist of seven-bit US-ASCII characters excluding SPACE, CTLS, and "`tspecials`" as defined in RFC-1521 [BORE93].
Encoding considerations:	The presence of binary data will typically require use of Content-Transfer-Encoding: BASE64
Security considerations:	See separate section in the document
Published specification:	Apple-single & Apple-double [APPL90]
Rationale:	Permits MIME-based transmission of data with Apple/Macintosh specific information, while allowing general access to non-specific user data.

4b. MULTIPART/APPLEDOUBLE

MIME type-name:	MULTIPART
MIME subtype name:	APPLEDOUBLE
Required parameters:	none
Optional parameters:	NAME, which must only consist of seven-bit US-ASCII characters excluding SPACE, CTLS, and "`tspecials`" as defined in RFC-1521 [BORE93].
Encoding considerations:	none
Security considerations:	See separate section in the document
Published specification:	Apple-single & Apple-double [APPL90]
Rationale:	Permits MIME-based transmission of data with Apple/Macintosh specific information, while allowing general access to non-specific user data.

4c. Detail specific to MIME-based usage

Macintosh documents do not always need to be sent in a special format. Those documents with well-known MIME types and non-existent or trivial resource forks can be sent as regular MIME body parts, without use of AppleSingle or AppleDouble.

Documents which lack a data fork should always be sent as AppleSingle.

All other documents should normally sent as AppleDouble. This includes documents with non-trivial resource forks, and documents without corresponding well-known MIME types.

It may be valuable in some cases to allow the user to choose one format over another, either because he disagrees with the implementor's definition of "trivial" resource forks, or for reasons of his own.

5. AppleSingle

An AppleSingle, version 2 file, is sent as one consecutive stream of bytes. The format is described in [APPL90] with a brief summary in Appendix A. The one and only part of the file is sent in an **application/applefile** message.

The first four bytes of an AppleSingle header are, in hexadecimal: 00, 05, 16, 00.

The AppleSingle file is binary data. Hence, it may be necessary to perform a Content-Transfer-Encoding for transmission. The safest encoding is Base64, since it permits transfer over the most restricted channels.

Even though an AppleSingle file includes the original Macintosh filename, it is recommended that a name parameter be included on the Content-Type header to give the recipient a hint as to what

file is attached. The value of the name parameter must consist of seven-bit US-ASCII characters excluding SPACE, CTLS, and "`tspecials`" as defined in RFC-1521 [BORE93].

5a. AppleSingle example

```
Content-Type: application/applefile; name="Computers-1/2-93"

    [The AppleSingle file goes here]
```

6. AppleDouble

An AppleDouble, version 2, file is divided in two parts:

Header: including the Macintosh resource fork and desktop information and
Data fork: containing the Macintosh data fork.

The AppleDouble format is described in [APPL90] with a brief summary in Appendix B.

The AppleDouble file itself is sent as a **multipart/appledouble** MIME body-part, which may have only two sub-parts. The header is sent as **application/applefile** and the data fork as whatever best describes it. For example, is the data for is actually a GIF image, it should be sent as **image/gif**. If no appropriate Content-Type has been registered for the data type, it should be sent as an **application/octet-stream**.

The first four bytes of an AppleDouble header are, in hexadecimal: 00, 05, 16, 07.

The AppleDouble header is binary data. Hence, it may be necessary to perform a Content-Transfer-Encoding for transmission. The safest encoding is Base64, since it permits transfer over the most restrictive channels.

Even though an AppleDouble file includes the original Macintosh filename, it is recommended that a name parameter be included on the Content-Type header of both the header and data parts of the AppleDouble file to give the recipient a hint as to what file is attached. The value of the name parameter must consist of seven-bit US-ASCII characters excluding SPACE, CTLS, and "`tspecials`" as defined in RFC-1521 [BORE93].

6a. AppleDouble example

```
Content-Type: multipart/appledouble; boundary=mac-part

--mac-part
Content-Type: application/applefile; name="My-new-car"
```

```
    [The AppleDouble header goes here]

--mac-part
Content-Type: image/gif;

    [The data fork goes here]

--mac-part--
```

7. References

BORE93 Borenstein N., and N. Freed, MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies, RFC 1521, Bellcore, Innosoft, September 1993.

APPL90 AppleSingle/AppleDouble Formats for Foreign Files Developer's Note, Apple Computer, Inc., 1990

8. Security Considerations

To the extent that **application/applefile** facilitates the transmission of operating-system sensitive data, it may open a door for easier relaxation of security rules than is intended either by the sender of the administrator of the sender's system.

9. Acknowledgements

Thanks to all of the people on the ietf-822 list who have provided much meaningful input for this document. Some of them must though be remembered by name, because they have almost crushed my mailbox the last weeks with a very nice and interesting debate:

> Johan Berglund, Steve Dorner, David Gelhar, David Herron, Raymond Lau, Jamey Maze, John B. Melby, Jan Michael Rynning, Rens Troost, Peter Svanberg

10. Authors Addresses

Patrik Fältström
Department of Numerical Analysis and Computing Science
Royal Institute of Technology
S-100 44 Stockholm
Sweden

Email: paf@nada.kth.se

Dave Crocker
675 Spruce. Dr.
Sunnyvale, CA 94086
USA

Email: dcrocker@mordor.stanford.edu

Erik E. Fair
Engineering Computer Operations
Apple Computer Inc.

Email: fair@apple.com

Appendix A. The AppleSingle format

In the AppleSingle format, a file's contents and attributes are stored in a single file in the foreign file system. For example, both forks of a Macintosh file, the Finder information, and an associated comment are arranged in a single file with a simple structure.

An AppleSingle file consists of a header followed by one or more data entries. The header consists of several fixed fields and a list of entry descriptors, each pointing to a data entry. Each entry is optional and may or may not appear in the file.

AppleSingle file header:

Field	Length
Magic number	4 bytes
Version number	4 bytes
Filler	16 bytes
Number of entries	2 bytes

Entry descriptor for each entry:

Entry ID	4 bytes
Offset	4 bytes
Length	4 bytes

Byte ordering in the file fields follows MC68000 conventions, most significant byte first. The fields in the header file follow the conventions described in the following sections.

Magic number

This field, modelled after the UNIX magic number feature, specifies the file's format. Apple has defined the magic number for the AppleSingle format as $00051600 or 0x00051600.

Version number

This field denotes the version of AppleSingle format in the event the format evolves (more fields may be added to the header). The version described in this note is version $00020000 or 0x00020000.

Filler

This field is all zeros ($00 or 0x00).

Number of entries

This field specifies how many different entries are included in the file. It is an unsigned 16-bit number. If the number of entries is any number other than 0, then that number of entry descriptors immediately follows the number of entries field.

Entry descriptors

The entry descriptor is made up of the following three fields:

Entry ID: an unsigned 32-bit number, defines what the entry is. Entry IDs range from 1 to $FFFFFFFF. Entry ID 0 is invalid.

Offset: an unsigned 32-bit number, shows the offset from the beginning of the file to the beginning of the entry's data.

Length: an unsigned 32-bit number, shows the length of the data in bytes. The length can be 0.

Predefined entry ID's

Apple has defined a set of entry IDs and their values as follows:

Data Fork	1 Data fork
Resource Fork	2 Resource fork
Real Name	3 File's name as created on home file system
Comment	4 Standard Macintosh comment
Icon, B&W	5 Standard Macintosh black and white icon
Icon, Colour	6 Macintosh colour icon
File Dates Info	8 File creation date, modification date, and so on
Finder Info	9 Standard Macintosh Finder information
Macintosh File Info	10 Macintosh file information, attributes and so on
ProDOS File Info	11 ProDOS file information, attributes and so on
MS-DOS File Info	12 MS-DOS file information, attributes and so on
Short Name	13 AFP short name
AFP File Info	14 AFP file, information, attributes and so on
Directory ID	15 AFP directory ID

Apple reserves the range of entry IDs from 1 to $7FFFFFFF. The rest of the range is available for applications to define their own entries. Apple does not arbitrate the use of the rest of the range.

Appendix B. The AppleDouble format

The AppleDouble format uses two files to store data, resources and attributes. The AppleDouble Data file contains the data fork and the AppleDouble Header file contains the resource fork.

The AppleDouble Data file contains the standard Macintosh data fork with no additional header. The AppleDouble Header file has exactly the same format as the AppleSingle file, except that it does not contain a Data fork entry. The magic number in the AppleDouble Header file differs from the magic number in the AppleSingle Header file so that an application can tell whether it needs to

look in another file for the data fork. The magic number for the AppleDouble format is $00051607 or 0x00051607.

The entries in the AppleDouble Header file can appear in any order; however, since the resource fork is the entry that is most commonly extended (after the data fork), Apple recommends that the resource fork entry to be placed last in the file. The data fork is easily extended because it resides by itself in the AppleDouble Data file.