INTERNET-DRAFT
draft-costanzo-fs-mime-00.txt
Expires: March 1st, 1999

A. Costanzo
AKC Computer Services Corp.

August 24, 1998

Creating File System Object Attachments with MIME

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 as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet- Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast, or ftp.isi.edu (US West Coast).

Distribution of this memo is unlimited.

2. Abstract

This memo defines a new top level type for MIME and its subtypes and mappings to file types.

The intent of this draft is to both define a new comprehensive top level type and define a method and usage for one of the to be defined subtypes. This new subtype in itself is a media type. The purpose of which provides a mechanism to move a structured set of files system object structures (directories and/or files) as a MIME attachment from one environment to another while preserving common elements.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

3. Introduction

In creating the conceptual aspects of this draft it was realized that a new top level media type needed to be defined.

Initial thoughts were to create a subtype in the Application top level type. This however seemed not only insufficient but inapplicable as well, since the type to be defined was in fact a media type.

Consideration was given to the creation of new top level media type for the specific objects in question, as to define these object types.

MIME's requirement for top level media types to have subtypes precluded this concept. The media type that was to be defined was in itself, self contained and had no such requirement for subtype definition. This in itself presented a dilemma.
How could this media type structure be defined within MIME?

There is in effect a large set of well defined types in a single name space that desperately need to be registered. This name space is "FILE".

This top level media type contains some abstract subtypes that have no definitive mapping, whereas others are quite definitive but do not fall well into any current MIME TYPE/subtype. An example of the former being "filename.txt" and an example of the latter being "filename.avi".

Defining this top level media type resolves the problem of this draft's subtype in question, "FS". FS as defined in this draft, is a media subtype object of the top level type "FILE".

This first draft makes absolutely no attempt to define all the subtypes in FILE. Nor does it attempt to deal with any abstract types as of yet. Let it currently be sufficient to say that FS is a media subtype of an newly defined top level type know as "FILE".

The FILE media type is not an attempt to circumvent usage of the other top level MIME types. FILE is in effect supplemental in nature.
Types may or may not exist in the other top level types and also exist in FILE.

This draft therefore specifies and defines a specific subtype for FILE, namely FS and its file type mappings.

The FS (File System) media type specifies a mail part consisting of a file system objects structure.

4. The FILE/FS Media Type

4.1 Overview

The Media Type FS specifies a section or part consisting of a file system media type object. File system media types provide a standard, transportable delivery system from many different operating systems.

The intent is to move a structured set of files from one environment to another while preserving common elements. At the same time, files can be moved within a single environment while preserving all attributes.

4.2 Definition of the FS Media Type

A FS media type object is a file system object that contains a representation of a file system object or objects.

FS media type objects are actually compressed file system object structures. They may be considered as a representation of a file, file structure, directory, directory and its contents and any other file system object that may be supported by a particular disk format. The structure may be thought of as a file archive that would be generated by a program like pkZip or ARJ - the difference being that the actual structure is stored in a format that will survive mail transport without any further modification or transfer encoding being applied.

4.3 MIME FS Media Type to File Name Mappings

The MIME FS media type always maps to a file with an extension or file hook of `.fs'. Implementors must use the `.fs' file extension. Mail agents that create a FS media type object by some internal functionality, such as drag and drop, should use the file name `default.fs' or some other temporary file name ending with the required `.fs' file hook.

4.4 Content-Type-Encoding

FS Media type objects have already been encoded and compressed using an encoding algorithm, with the exception of control elements known as "sections". The Content-Type-Encoding for this media type therefore must always be "none". A mailer must not do any additional encoding to a part containing a FS Media type object.

In other words Content-Type-Encoding must be ignored or the FS object will no longer be in a valid format.

4.5 Character Set

FS Media type objects are in UTF-8 character set. [2] [3]
This character set must be used.

4.6 Partial or Split Message Parts and Message Boundaries

A single FS media type object must be completely contained in a specific message part and must not be split over multiple parts in any fashion. Only one FS media object may be contained in a message part or boundary.

If it is desirous to have a split FS object, at least one instance of the complete object must be contained within a message part.

4.7 Content Disposition

Content Disposition of a FS Media Type object must always be set to `attachment'.

4.8 An example of an FS Media Type Object sent via a MIME program:

```
        Content-Type: multipart/mixed;
              boundary="----=_NextPart_000_0099_01BDC836.52F1A800"

        This is a multi-part message in MIME format.

        ------=_NextPart_000_0099_01BDC836.52F1A800
        Content-Type: text/plain;
            charset="utf-8"
        Content-Transfer-Encoding: 7bit

       Hi! I'm sending you my windows directory, it's attached.
        -----------------------------------------------------

        And Remember --- No Matter Where You Go, There You Are...
```

```
 ------=_NextPart_000_0099_01BDC836.52F1A800
Content-Type: FILE/fs;
      charset="utf-8"
      name="default.fs"
Content-Disposition: attachment;
     filename="default.fs"
      [ directory windows
      created 27 Mar 1996 12:17:20.09
      modified 1 Aug 1998 11:31:04.00
      [file window~1
      display "windows Long Filename"
      created 1 Aug 1998 11:31:05.09
      modified 1 Aug 1998 11:31:05.09
      type FLAT
      [ data LZJU90
      * LZJU90
      ...
      ]]]
```

```
------=_NextPart_000_0099_01BDC836.52F1A800--
```

5. Creating FS Media Types

As mentioned previously, FS media type objects are actually compressed file system object structures stored as a compressed file.

They may be considered as a representation of a file, file structure, directory, directory and its contents and any other file system object that may be supported by a particular disk format. The structure may be thought of as a file archive that would be generated by a programs like pkZip, ARJ or Gzip, the difference being that the actual structure is stored in a format that will survive mail transport.

The FS media type may be created by a standalone encoder / decoder or any mail program such as Microsoft Outlook Express. FS encoding creates these type objects. See section 5.2.

Programs and mailers have been using a version of the FS since the mid 1980's. FS was previously defined in RFC 1505 [1]. It is redefined in this Draft.

When using a standalone FS encoder program it would be possible to insert the raw data created by the encoder into a mail message, send it to someone with a decoder and successfully reconstruct the file system object(s).

MIME mail users must first use a file archive program like Gzip or manually attach file after file to their email. Even more complex is when it is desirous to send complex directory structures.

FS encoding solves this by allowing an FS object to be attached directly to an electronic mail messages.

5.2 Creating a FS Object using FS Encoding

5.2.1 Overview

As previously stated, the intent is to move a structured set of files from one environment to another while preserving common elements. At the same time, files can be moved within a single environment while preserving all attributes.

The representations consist of a series of nested sections, with attributes defined at the appropriate levels. Each section begins with an open bracket "[" followed by a directive keyword and ends with a close bracket "]". Attributes are lines, beginning with a keyword. Lines which begin with a LWSP (linear white space) character are continuation lines.

Any string-type directive or attribute may be a simple string not starting with a quotation mark (") and not containing special characters (e.g. newline) or LWSP (space and tab). The string name begins with the first non-LWSP character on the line following the attribute or directive keyword and ends with the last non-LWSP character.

Otherwise, the character string name is enclosed in quotes. The string itself contains characters in ISO-10646-UTF-8 but is quoted and escaped at octet level (as elsewhere in RFC822). The strings begin and end with a quotation mark ("). Octets equal to quote in the string are escaped, as are octets equal to the escape characters (\" and \\). The escaped octets may be part of a UTF multi-octet character. Octets that are not printable are escaped with \nnn octal representation. When an escape (\) occurs at the end of a line, the escape, the end of the line, and the first character of the next line, which must be one of the LWSP characters, are removed (ignored).

```
 [ file Simple-File.Name
 [ file "   Long file name starting with spaces and having a couple\
   [sic] of nasties in it like this newline\012near the end."
```

Note that in the above example, there is one space (not two) between "couple" and "[sic]". The encoder may choose to use the nnn sequence for any character that might cause trouble.

5.3 Sections

A section starts with an open bracket, followed by a keyword that defines the type of section.

   The section keywords are:

```
             directory
             entry
             file
             segment
             data
```

The encoding may start with either a file, directory or entry. A directory section may contain zero or more file, entry, and directory sections. A file section contains a data section or zero or more segment sections. A segment section contains a data section or zero or more segment sections.

5.3.1 Directory

This indicates the start of a directory. There is one parameter, the entry name of the directory:

```
             [ directory foo
             ...
             ]
```

5.3.2 Entry

The entry keyword represents an entry in a directory that is not a file or a sub-directory. Examples of entries are soft links in Unix, Microsoft Windows shortcuts or access categories in Primos.
A Primos access category might look like this:

```
             [ entry SYS.ACAT
             type ACAT
             created 27 Jan 1987 15:31:04.00
             acl SYADMIN:* ARIEL:DALURWX $REST:
             ]
```

5.3.3 File

The file keyword is followed by the entry name of the file. The section then continues with attributes, possibly segments, and then data.

```
             [ file MY.FILE
             created 27 Feb 1987 12:10:20.07
             modified 27 Mar 1987 16:17:03.02
             type DAM
             [ data LZJU90
             * LZJU90
             ...
             ]]
```

5.3.4 Segment

This is used to define segments of a file. It should only be used when encoding files that are actually segmented. The optional parameter is the number or name of the segment.

When encoding Macintosh files, the two forks of the file are treated as segments:

```
             [ file A.MAC.FILE
             display "A Mac File"
             type MAC
             comment "I created this myself"
             ...
             [ segment resource
             [ data ...
             ...
             ]]
             [ segment data
             [ data ...
             ...
             ]]]
```

5.3.5 Data

The data section contains the encoded data of the file. The encoding method shown is defined in a separate Internet Draft. The data section must be last within the containing section.

5.4 Attributes

Attributes may occur within file, entry, directory, and segment sections. Attributes must occur before sub-sections.

The attribute directives are:

```
             display
             type
             created
             modified
             accessed
             owner
             group
             acl
             password
             block
             record
             application
```

5.4.1 Display

This indicates the display name of the object. Some systems, such as the Macintosh, use a different form of the name for matching or uniqueness. Microsoft Windows 95/98 use the attribute to maintain the long filename.

5.4.2 Comment

This contains an arbitrary comment on the object. The Macintosh stores this attribute with the file.

5.4.3 Type

The type of an object is usually of interest only to the operating system that the object was created on.

Types are:

ACAT	access category (Primos)
CAM	contiguous access method (Primos)
DAM	direct access method (Primos)
FIXED	fixed length records (VMS)
FLAT	`flat file', sequence of bytes (Unix, DOS, default)
ISAM	indexed-sequential access method (VMS)
LINK	soft link (Unix), Microsoft Windows shortcut
MAC	Macintosh file
SAM	sequential access method (Primos)
SEGSAM	segmented direct access method (Primos)
SEGDAM	segmented sequential access method (Primos)
TEXT	lines of ISO-10646-UTF-8 text ending with CR/LF
VAR	variable length records (VMS)

5.2.4 Created

Indicates the creation date of the file. Dates are in the format defined in section 5.3.

5.2.5 Modified

Indicates the date and time the file was last modified or closed after being open for write.

5.2.6 Accessed

Indicates the date and time the file was last accessed on the original file system.

5.2.7 Owner

The owner directive gives the name or numerical ID of the owner or creator of the file.

5.2.8 Group

The group directive gives the name(s) or numerical IDs of the group or groups to which the file belongs.

5.2.9 ACL

This directive specifies the access control list attribute of an object (the ACL attribute may occur more than once within an object). The list consist of a series of pairs of IDs and access codes in the format:

```
user-ID:access-list
```

There are four reserved IDs:

$OWNER	the owner or creator
$GROUP	a member of the group or groups
$SYSTEM	a system administrator
$REST	everyone else

The access list is zero or more single letters:

A	add (create file)
D	delete
L	list (read directory)
P	change protection
R	read
U	use
W	write
X	execute
*	all possible access

5.2.10 Password

The password attribute gives the access password for this object. Since the content of the object follows (being the raison d'etre of the encoding), the appearance of the password in plain text is not considered a security problem. If the password is actually set by the decoder on a created object, the security (or lack) is the responsibility of the application domain controlling the decoder as is true of ACL and other protections.

5.2.11 Block

The block attribute gives the block size of the file as a decimal number of bytes.

5.2.12 Record

The record attribute gives the record size of the file as a decimal number of bytes.

5.2.13 Application

This specifies the application that the file was created with or belongs to. This is of particular interest for Macintosh files.

5.3 Date Field

Various attributes have a date and time subsequent to and associated with them.

5.3.1 Syntax

The syntax of the date field is a combination of date, time, and timezone:

```
    DD Mon YYYY HH:MM:SS.FFFFFF [+-]HHMMSS

    Date :=  DD Mon YYYY      1 or 2 Digits " " 3 Alpha " " 4 Digits
    DD   :=  Day              e.g. "08", " 8", "8"
    Mon  :=  Month            "Jan" | "Feb" | "Mar" | "Apr" |
                              "May" | "Jun" | "Jul" | "Aug" |
                              "Sep" | "Oct" | "Nov" | "Dec"
    YYYY :=  Year
    Time :=  HH:MM:SS.FFFFFF  2 Digits ":" 2 Digits [ ":" 2 Digits
                                 ["." 1 to 6 Digits ] ]
                                 e.g. 00:00:00, 23:59:59.999999
    HH   :=  Hours            00 to 23
    MM   :=  Minutes          00 to 59
    SS   :=  Seconds          00 to 60 (60 only during a leap second)
    FFFFF:=  Fraction
    Zone :=  [+-]HHMMSS       "+" | "-" 2 Digits [ 2 Digits
                                 [ 2 Digits ] ]
    HH   :=  Local Hour Offset
    MM   :=  Local Minutes Offset
    SS   :=  Local Seconds Offset
```

5.3.2 Semantics

The date information is that which the file system has stored in regard to the file system object. Date information is stored differently and with varying degrees of precision by different computer file systems. An encoder must include as much date information as it has available concerning the file system object.

A decoder which receives an object encoded with a date field containing greater precision than its own must disregard the excessive information. Zone is Co-ordinated Universal Time "UTC" (formerly called "Greenwich Mean Time"). The field specifies the time zone of the file system object as an offset from Universal Time. It is expressed as a signed [+-] two, four or six digit number.

A file that was created April 15, 1993 at 8:05 p.m. in Roselle Park, New Jersey, U.S.A. might have a date field which looks like:

15 Apr 1993 20:05:22.12 -0500

5.0 Notes to Implementors

This section of this draft no way attempts to specify any type of standard. It is merely here as a reference to implementors. NONE of the following section is required to implement the FS media type.

5.1 Inter-operablility with non-MIME mail systems and stand alone Encoders/Decoders

Standalone FS encoder and decoding programs will be able to decode a FS object sent via electronic mail regardless if that mailer is MIME compliant. The mailer only need be able to support the UTF-8 character set. The mail section with the FS object will need to be cut from the messages beginning at the first section control element.

Mail messages from a MIME compliant mailers will be successfully translated by non-MIME mailers directly without the need of a standalone decoder that uses an encoding header field but only if the MIME implementation includes the field as well in the mail message, otherwise a standalone encoder/decoder will need to be used.

6. Security Consideration

The security (or lack) is responsibility of the application domain controlling the decoder of a FS media type object.

7. References

[1] Costanzo, A. Robinson, D. and R. Ullmann, "Encoding Header Field for Internet Messages", RFC 1505, AKC Consulting, Prime Computer, Inc., August 1993.

[2] International Organization for Standardization, Information Technology -- Universal Coded Character Set (UCS). ISO/IEC 10646-1:1993, June 1993.

[3] International Organization for Standardization, Information Technology -- Universal Coded Character Set (UCS). ISO/IEC 10646-1: 1993/AMD.2: 1996 (E)

8. Epilogue

It should be noted that the author of this Draft did not participate in the initial design of FS. Intentions exist to create a plug-in implementation for various MIME compliant mailers. The plug-in will allow the transfer of file system directory structures.

A definition of the encoding mentioned in this draft may be found in a separate draft titled: "Definition of the LZJU90 MIME Content Transfer Encoding Type".

The author is attempting to solicit comments, suggestions and recommendations from the Internet community in general.

You may join the File System Attachment discussion list by sending email to: "fsa-request@donet.com", the body of the message must consist of the word: "subscribe".

9. Acknowledgments

The author would like to thank Robert Jung and David Robinson and Robert Ullmann for their past contributions to this work as well as all of the people who have made suggestions to this current project.

10. Authors' Addresses

Al Costanzo
AKC Computer Services Corp.
P.O. Box 4031
Roselle Park, NJ 07204-0531

Phone: +1 908 298 9000
Email: AL@AKC.COM