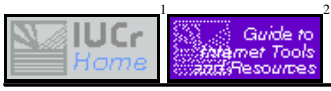

E–mail and multi–media extensions

E–mail services are well–known to and much–used by academic users having been introduced through the BITNET and UUCP networks well over ten years ago. In general these, and some ISO/X400 systems experimented in Europe, have given way to Internet e–mail, a basic service which has doubtless had an enormous impact on communication within the academic and research community due to its simplicity and robustness.

When a user sends an e–mail by the internet, the message is despatched using the *Simple Mail Transfer Protocol* (SMTP) for transmission by direct communication to a local SMTP server machine. On the other hand, incoming messages are stored on a local 'mail–box' server from which the user can retrieve his e–mails when activating his e–mail program (known as a UA – User Agent). Various protocols for the transmission of e–mails between the mail–box server and the UA are in operation, of which the most common are POP, *Post Office Protocol* and IMAP, *Interactive Mail Access Protocol* or *Internet Message Access Protocol*. Other more advanced protocols are in development.

However the basic Internet e–mail is limited to textual messages of restricted line length, character set and message length. Of course ways round these restrictions can and have been improvised but their application tends to be rather irksome. Further these solutions are often platform–dependent. User expectation of e–mail systems has risen with the widespread introduction of graphical terminals and stations using windowing 'Graphical User Interfaces' and equipped with various multi–media hardware. One needs to be able to send and receive long messages with lines of arbitrary length in any character set, executable programs, word processors files, drawings, photographs, audio and video clips, *etc.* in a manner as simple as putting a photograph in an envelope.

The 'Internet solution' to the above problems, developed with the necessities of multi–platform compatibility and interoperability in mind, was proposed by Nathaniel Borenstein under the name MIME – Multipurpose Internet Mail Enclosures. In fact MIME is not limited only to internet e–mail applications but is extensively used on the W3 for transferring its various information formats.

What is MIME?

Quoting from the comp.mail.mime FAQ[3] which is a source of reliable information on MIME:

MIME, the Multi–purpose Internet Mail Extensions, is a freely available specification that offers a way to interchange text in languages with different character sets, and multi–media e–mail among many different computer systems that use Internet mail standards.

If you were bored with plain text e–mail messages, thanks to MIME you now can create and read e–mail messages containing these things:

- *character sets other than ASCII*
- *enriched text*
- *images*
- *sounds*
- *other messages (reliably encapsulated)*
- *tar files*
- *PostScript*
- *pointers to FTPable files*
- *other stuff*

MIME supports not only several pre–defined types of non–textual message contents, such as 8–bit 8000Hz–sampled mu–LAW audio, GIF image files, and PostScript programs, but also

[1] http://www.iucr.ac.uk/iucr–top/welcome.html
[2] int.index.html
[3] http://www.cis.ohio–state.edu/text/faq/usenet/mail/mime–faq/top.html

permits you to define your own types of message parts.

The ability to create e–mail messages with audio and other non–textual contents has been around for a while, but almost always as part of a vendor–specific "solution." This means that you can't create a message on a NeXT system containing PostScript information and "Lip Service" (NeXT's audio e–mail tool) and easily handle the same message on an HP 9000/710, a Sun SPARCstation IPC, and a Silicon Graphics Iris. That's a problem that MIME helps to solve.

MIME – How it works

MIME defines a mechanism to declare the content of a message and the required transformations to transport it as though it were a simple text. For example, this approach enables files to be attached to e–mails with headers of the type 'this is Word document encoded with binhex', 'this is an image in GIF format',or simply 'this is a text which uses the ISO–8859–1 set of characters' added by the sender's UA and interpreted automatically by the receiver's UA. Between MIME User Agents these transformations and interpretations are transparent for the users.

MIME is simple so that it can be easily implemented on any system. MIME is entirely backward–compatible with the older and simpler internet e–mail protocols which only allow one to transfer text with ASCII characters and lines of limited length. MIME can be extended to allow new file formats to be transfered. Its authors designed MIME *to make multi–part, multi–character set, multi–media e–mail widely available on the Internet. MIME has been designed as a mechanism for encapsulating a wide variety of data types that have been found useful in different environments, and was explicitly designed for easy extension to include new formats that have been found practical in other contexts.*

MIME extensions place two new fields in the e–mail header allowing the definition of the content–type and coding.

Content–Transfer–Encoding

MIME defines precisely how messages are to be coded for transfer:

- **7bit** really means 'no coding' and is the default value when there is no Content–Transfer–Encoding field in the message header, the case for non–MIME systems.
- **8bit** also means 'no coding' and is useful for the transfer of messages between systems using the ISO–8859–1 (ISO–latin–1) character set with accentuated characters.
- **quoted–printable** uses only 7–bit ASCII characters and represents other characters by a sequence starting with a **=** followed by the hexadecimal value of the character. Further a **=** followed by a line–feed is inserted for lines longer than 76 characters.
- **base64** is a simple, dense but uncompressed coding system known to be reliable when crossing gateways (not the case with *uuencode*). [It is well known that gateways often degrade e–mail messages irreversibly due to different character sets, adding or deleting end–of–line marks, truncation of long lines, headers scrambled, long messages truncated, and more!] base64 is the way to encode binary files for 7–bit transport. The coded message wil be 33% larger than the original (3 binary bytes are coded as 4 characters from an alphabet of 64 7–bit characters.

One should avoid using non–standard or platform dependent encoding where the receiver UA will be unable to decode the message. Note however that a binary Macintosh file can be coded '7bit' but with a type 'application/mac–binhex40'. In this way, if the message is not degraded by a gateway, the receiver who does not have a MIME–compatible UA will be able to copy it to a file and decode it with BinHex4.0 whilst for a MIME–compatible UA the decoding will take place automatically.

Content–Type

MIME defines 7 types of content (in RFC 1521) to which are added sub–types and sometimes parameters. The full list of Content–Type definitions[4] is available from IANA (Internet Assigned

[4] ftp://isi.edu/in–notes/media–type/media–types

Numbers Authority). Here is a short description of the main 'type/subtype;parameter':

`text/plain;charset="iso-8859-1"`
: Text with accentuated characters using the ISO latin–1 set

`text/plain;charset="us-ascii"`
: Text using 7–bit ASCII. This is the format of messages from non–MIME UAs

`multipart/mixed;boundary="string"`
: Message in several independent parts which can have different content types. For example, the first part is a text file explaining how to use the program sent in the second part of the message.

`multipart/alternative;boundary="string"`
: Message formed of several versions of the same information. For example, the same text is sent as a simple text file followed by a Word file readable on a Macintosh or a pc, and finally a postscript file for printing or viewing.

`multipart/parallel;boundary="string"`
: A message for which all the parts can be viewed simultaneously. For example: a document with text, image and sound commentary.

`multipart/digest;boundary="string"`
: A message composed of several messages of text in format RFC822 each with their own header (From:, To:,Subject:). For example a condensed format of messages coming from a list server.

`message/partial;number=x;total=y`
: Part of a message too large to be transmitted in one part. The message is thus fragmented on despatch and reconstituted on reception, if possible automatically.

`message/external-body;access-type=anon-ftp`
: Instead of transmitting a file with the message, the method of access to this file is sent.

`application/postscript;`
: A postscript file which can be automatically viewed by the receiver UA. There is a security risk in this as the postscript language allows file operations on the target machine.

`application/mac-binhex40;name="string"`
: A message file coded in the Macintosh BinHex format. For example this might be an Excel or Word document sent as an Eudora attachment.

`image/jpeg;`
: Graphics file in JPEG format

`image/gif;`
: Graphics file in GIF format

`audio/basic;`
: Audio file in '8–bit ISDN mu–law'

`video/mpeg;`
: Video file in MPEG format

Apart from these standard types, registered and approved by IANA, other 'private' type and sub–types are also in existence and are generally only know to one type or series of platform *e.g.* `video/x-sgi-move` or `application/x-Framemaker`.

What are the limitations of MIME?

MIME is tied down by the necessity of backwards compatibility and thus has some limitations. A MIME User Agent can always receive a MIME message, almost always decode it but there is not guarantee that the message can be interpreted correctly. If you send an Excel file to a person who does not have Excel installed on his machine it is obvious that he can be nothing with it even if it is correctly transmitted and decoded.

Take the case where you send a message to a person who does not have a MIME–compatible UA. In the message there are lines which have more than 76 characters and some accentuated characters such as Åäóö. Your UA will probably code the message as 'quoted–printable' making it more–or–less readable but with some = sequences in it. If the message is encoded in 'base64' the message will be unreadable on reception.

A further limitation concerns the message header. Many e–mail programmes go haywire if they encounter accentuated characters in the e–mail header. MIME defines a coding method for

non–ASCII characters in the message header (RFC1522) but it is advisable to only use non–ASCII characters in the body of the message.

POP and IMAP protocols

To access e–mail in a mail–box server, there exist two families of protocols: POP = *Post Office Protocol* and IMAP *Interactive Mail Access Protocol* or *Internet Message Access Protocol*. POP is a simple protocol dating from February 1985 and can thus be found both on pcs and mainframes. With POP once the user has identified himself on the mail–box server, all new messages are transfer to the machine running his UA (store and forward to client). Connection is only maintained for the period of the transfer. On the other hand IMAP is more modern (August 1990) and is a much richer protocol. Over the functionality offered by POP, IMAP allows searching and selecting messages on the mail–box server explaining that in general messages are left on the mail–box server but with the overhead of a permanent connection between the mail–box and the UA during the session.

Documentation

An excellent source of documentation on many aspects of Internet e–mail is to be found in the usenet comp.mail.* newsgroups. In the FAQs (Frequently Asked Questions) of these newsgroups one finds the answers to the problems one is confronted with.

[Index][5] – 20th July 1996 – © Howard Flack[6] – **Not to be copied or reproduced without permission**

[5] int.index.html
[6] ../ahdf/Howard.Flack.html