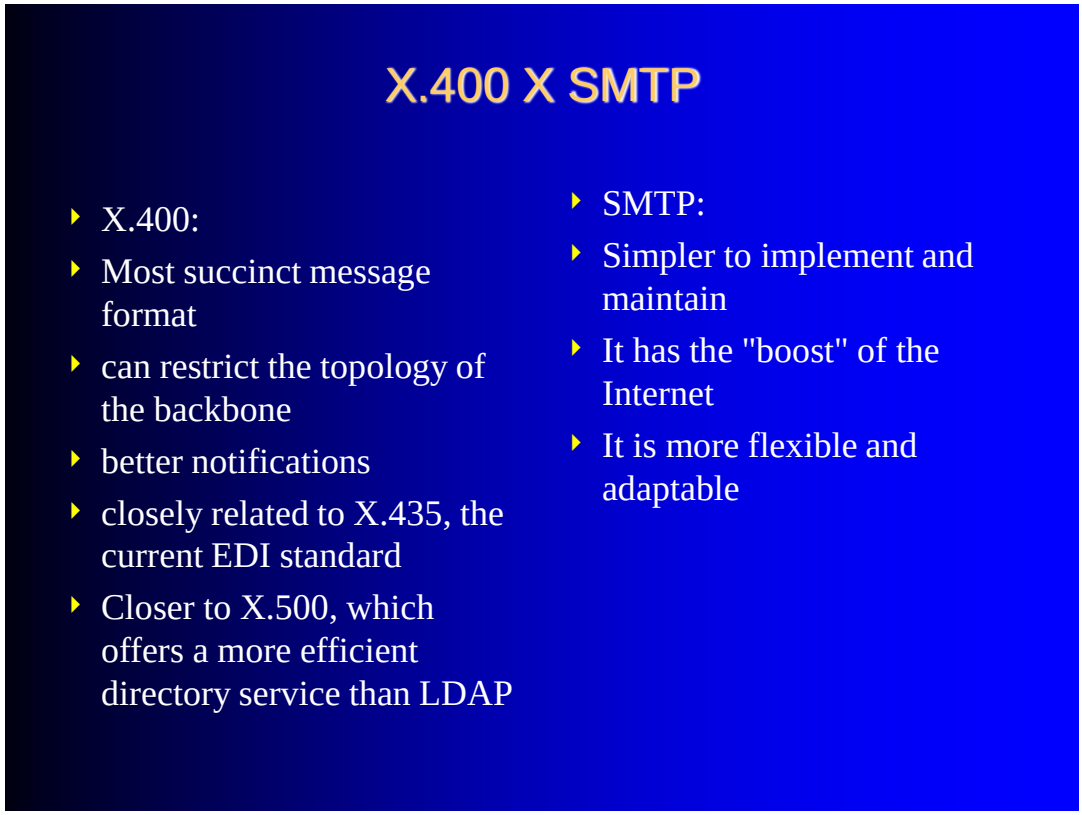
Workshop AMHS 1

2023

Francisco Almeida – SAM CNS Officer



Comparisons between X.400 and SMTP systems

It is worth mentioning the inevitable comparison that is made between the two deployed backbones of wider use: the SMTP of the Internet and the X.400/MOTIS. The Internet messaging service is admittedly regarded as more widely disseminated in terms of connected users. However, those users and applications that involve a "mission critical" aspect, which includes air traffic management, generally tend toward the use of MHS. John Rothon lists some technical advantages of X.400/MOTIS over SMTP [Rothon97]:

- Offers a succinct message format;

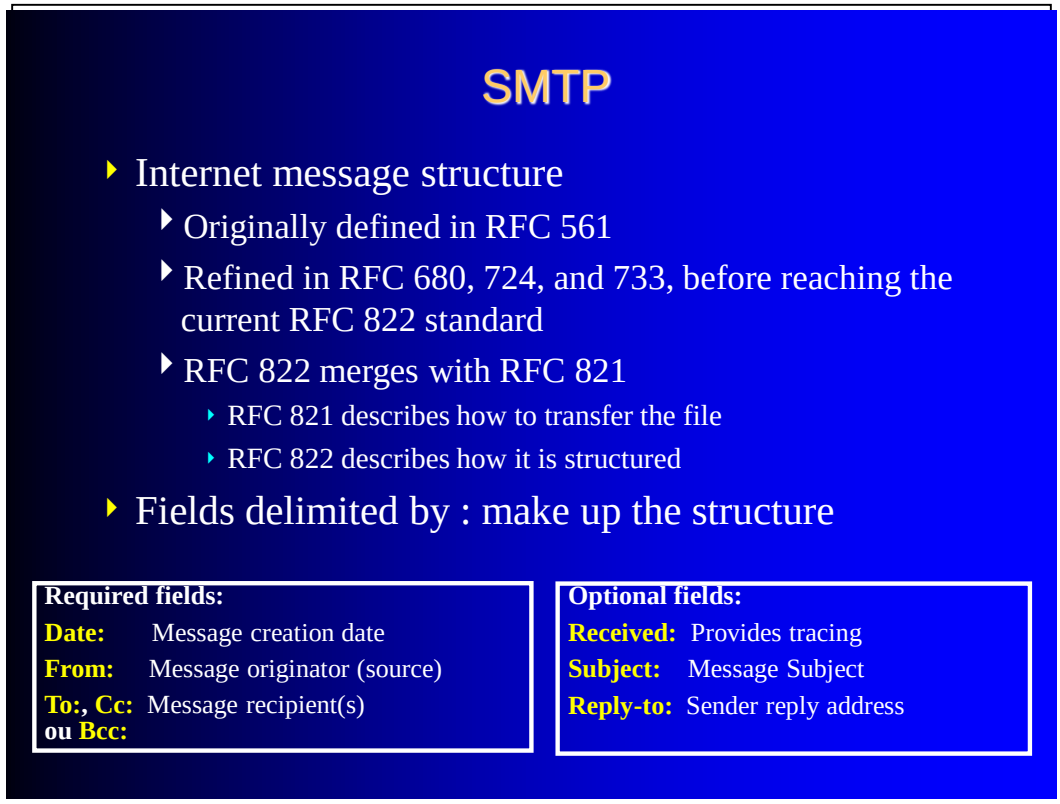
- Allows you to restrict the topology of the message backbone;

- Offers better notifications;

- It is closely affiliated with X.435 which is the current EDI standard; and,

- X.400 is more closely related to X.500 which offers a more comprehensive Directory Service than LDAP.

In a survey conducted by the Electronic Messaging Association - EMA in 1997, seventy-four percent of managers and system administrators interviewed indicated a preference for products based on X.400 recommendations in order to achieve better quality levels for their business [Rubenstein98].



RFC 822 (SMTP) Structure

On the Internet, the standard format for encapsulating a message is described in RFC 822. This Request for Comment (RFC) is used in conjunction with RFC 821, although the two are independent of each other.

Each row of the header consists of a field (terminated by a colon). The field normally fits on one line, but if it is too long, it can be expressed on multiple lines. Each continuation line begins with a space simply.

There are only a few required fields to have a valid message. The date (time it is created), the source (sender) and the recipient of the message are required.

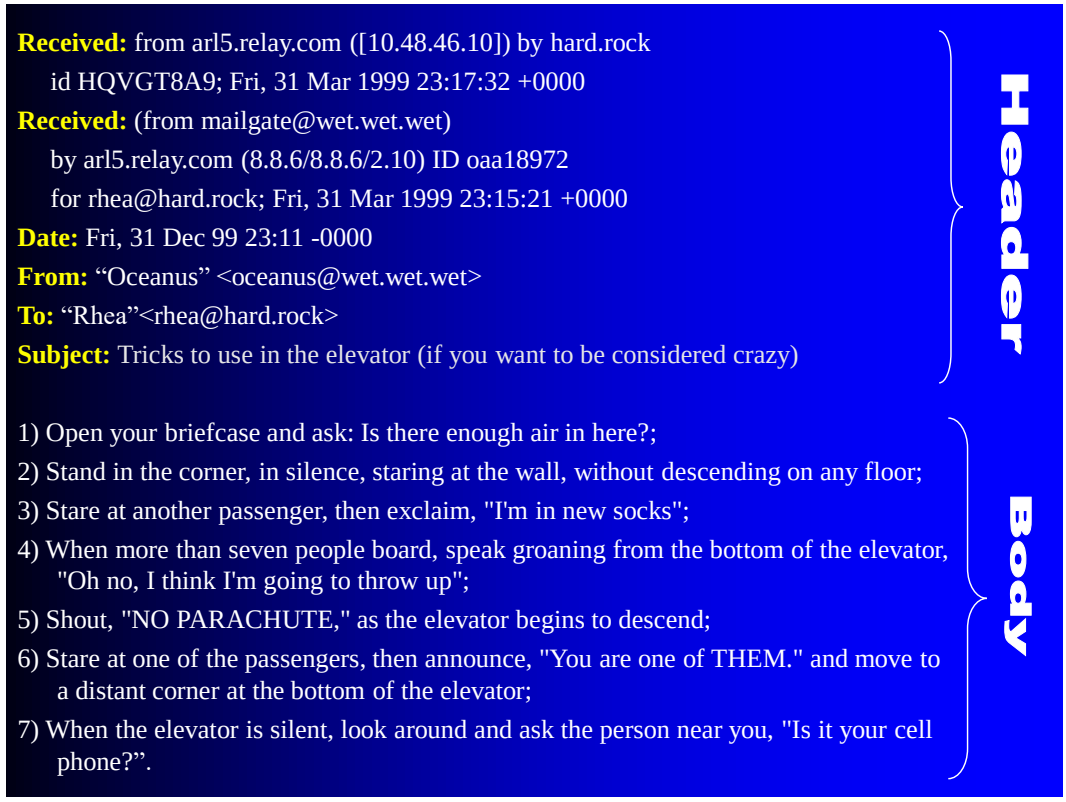
A message consists of a header and a body separated by a blank line (two consecutive CR and LF). Since blank lines are not allowed in the header, the end of the header section and the beginning of the body are clearly demarcated.

HEADER SECTION

Each row of the header consists of a label, followed by a colon and a value. In most cases, one line is enough. However, multiple lines can be used, as long as they start with space.

SYNTAX

Each default field is defined in RFC 822 using the Backus-Naur Form. The syntax of some fields is free (such as those of Subject, Comments, and Keywords). Attention should be paid to the syntax of two fields, those that express dates and addresses.



DATES:

[week day “,”] day moth year hours “:” minutes [“:”seconds] zone, as below:

Fri, 31 Dec 99 23:04:11 -0000

- The day of the week is made up of the first three letters of the days in English (Mon, Tue, Wed, Thu, Fri, Sat and Sun), whatever the language of the users.

- The same applies to the months (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov and Dec).

- It is obvious that RFC 822 predates the questioning of the year 2000 (Y2K). The original specification (which was updated in RFC 1123) adopts only two digits for the year. Fortunately, this will only cause ambiguity around the year 2082.

- The time is simple. It should be noted that it uses values referring to 24 hours.

- The time zone can be one of the following: UT, GMT, EST, EDT, CST, MST, MDT, PST and PDT; or, preferably, the four-digit GMT (Greenwich Mean Time) off-set.

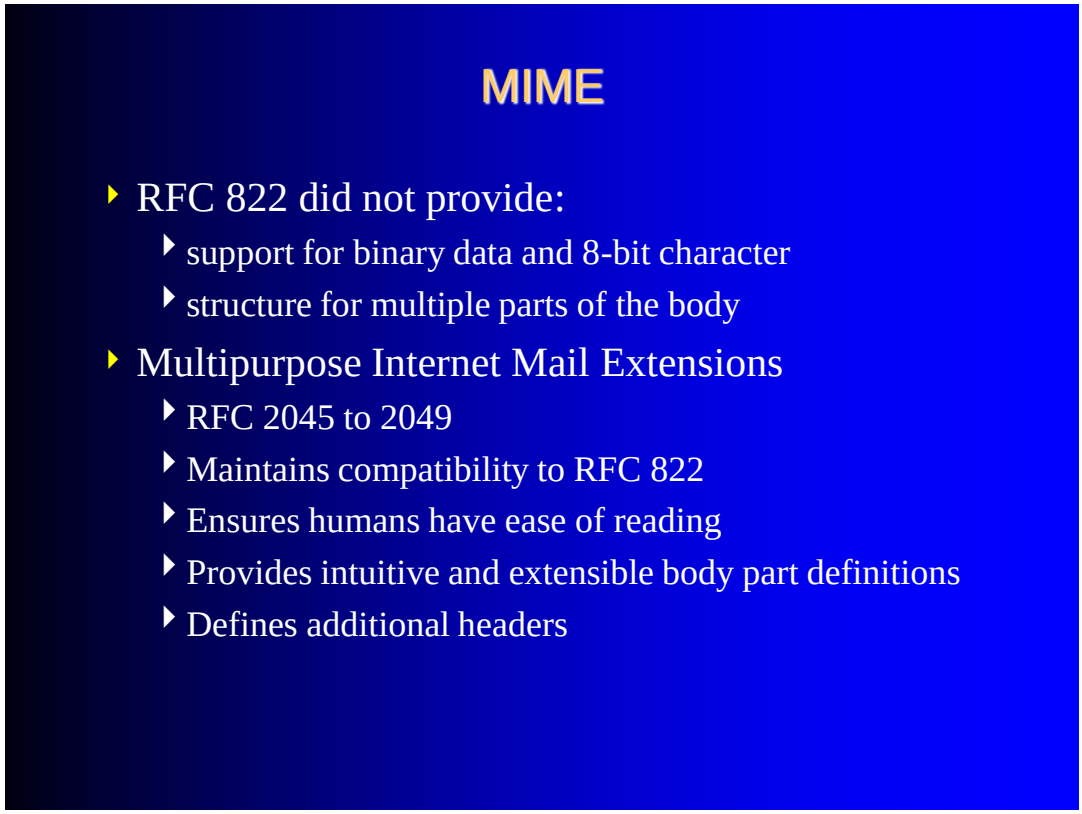
ADDRESSES:

Addresses can be specified in simple or expanded forms. The simple form uses the typical format:

Mail-box@domain-name-fully-qualified. For instance: almeida@cesupa.br

In the expanded form, one can attach a "friendly" description to the address:

Friendly-name <simple-address>. For instace: Francisco Almeida <almeida@cesupa.br>



RFC 822 had two flaws:

- Lack of support for binary data and 8-bit characters;

- Lack of a structure for multiple parts of the body.

To work around these shortcomings, the Multipurpose Extensions for Internet Mail (MIME), defined in RFC 2045 through 2049, were created. The people who developed MIME, in addition to creating a format without these restrictions, tried to accommodate other considerations such as:

- Maintain backward compatibility with RFC 822;

- Ensure maximum readability for humans; and

- Provide intuitive and extensible body part definitions.

MIME is implemented as an extension of RFC 822. Instead of creating a new format, RFC 822 is used with the following enhancements:

- Defines additional header fields using the standard syntax of RFC 822, which are ignored by non-MIME readers;

- Defines a structure for the message body. Non-MIME readers will not be able to "decipher" the body of the message demarcated by the MIME. Typically, a preamble is placed with a warning for those who do not have MIME support.

Since RFC 822 admits only 7-bit characters, with MIME two types of format encoding were introduced that converts 8-bit data into 7-bit characters:

- Base64 and QuotedPrintable.

```
Date: Fri, 31 Dec 99 23:59 -0000
From: "Almeida" <almeida@cesupa.br>
To: "Nilton" <cns2-2@decea.gov.br>
Subject: Teste
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="--Boundary. 9241449.2213818--"

Esta mensagem tem o formato MIME. Se o seu programa de correio não tem suporte
para MIME, você não conseguirá lê toda a mensagem.

----Boundary.9241449.2213818--
Prezado Sr.,
Adjunto el archive lista.txt

----Boundary.9241449.2213818--
Content-Type: text/plain;
        name="lista.txt"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: attachment;
        filename="lista.txt"

Marcelo Ure=F1a, Cl=E9ment Chevallier, Ant=F4nio Oliveira ...
----Boundary.9241449.2213818------
```

The first indication that the message has MIME format is the presence of additional fields. One of them is the Content-Type that specifies the limit (Boundary).

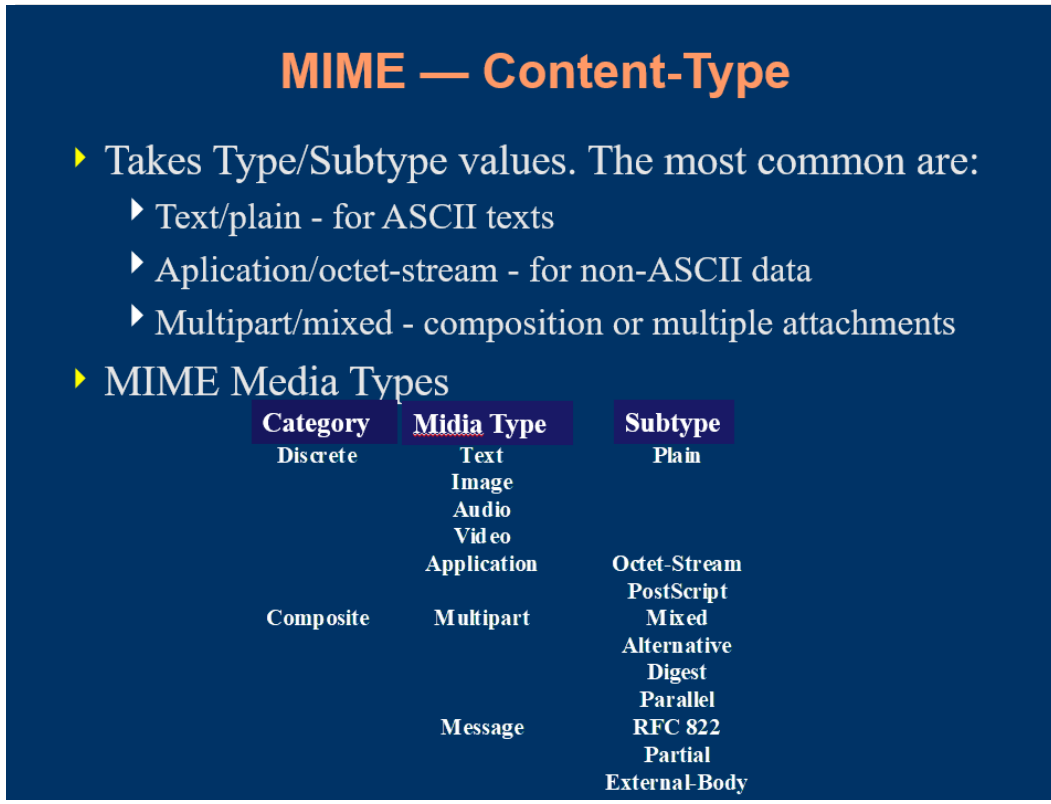
After the blank line, MIME subdivides the body into two sections. Up to the first limit is the preamble, which simply communicates to users without MIME support that they may have trouble viewing the full message. MIME users don't see this preamble.

After each boundary is an Attachment. Thus it continues until the final limit, which is preceded by two dashes and terminated by two dashes. After the final limit comes the epilogue (optional) which is also ignored by MIME readers.

As mentioned earlier, MIME makes use of additional headers. In the table below you can see five of these:

Label	**Tipical Format**
MIME-Version	Number.Number
Content-ID	Message ID
Content-Descriptor	Free text
Content-Type	Type/subtype; parameters
Content-Transfer-Encoding	Encoding mechanism

The MIME-Version is very simple and is in version 1.0. The following two are optional and need no further explanation. Content-Type allows you to structure the message by declaring that it is of a particular type, or alternatively, that it is composed of multiple parts. Content-Transfer-Encoding allows you to encode data from 8 bits to 7 bits, or leave it as it is, if it is already 7 bits.



Content-Type

The idea of using media types is so that the recipient's reader (software) knows what kind of data was sent, so that it can associate it with the correct application.

Content-Transfer-Encoding

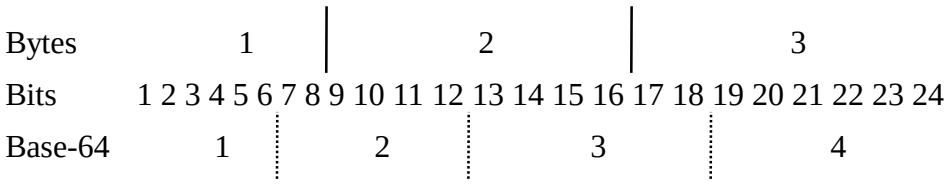
This field describes what mechanism is used to convert 8-bit data to the 7-bit format. Two widely used mechanisms are described below:

Base64

So called because it uses only 64 characters (10 decimal digits, 26 uppercase letters, 26 lowercase letters, "+" and "/". Coding has two processes:

- Convert 8-bit characters to 6-bit characters.

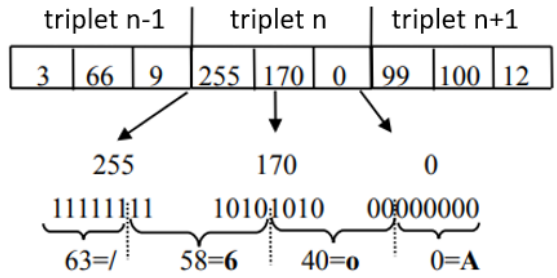
- Map the 6-bit values in the Base64 character set.



Base-64 Encoding Table

0 A	16 Q	32 g	48 w
1 B	17 R	33 h	49 x
2 C	18 S	34 i	50 y
3 D	19 T	35 j	51 z
4 E	20 U	36 k	52 0
5 F	21 V	37 l	53 1
6 G	22 W	38 m	54 2
7 H	23 X	39 n	55 3
8 I	24 Y	40 o	56 4
9 J	25 Z	41 p	57 5
10 K	26 a	42 q	58 6
11 L	27 b	43 r	59 7
12 M	28 c	44 s	60 8
13 N	29 d	45 t	61 9
14 O	30 e	46 u	62 +
15 P	31 f	47 v	63 /

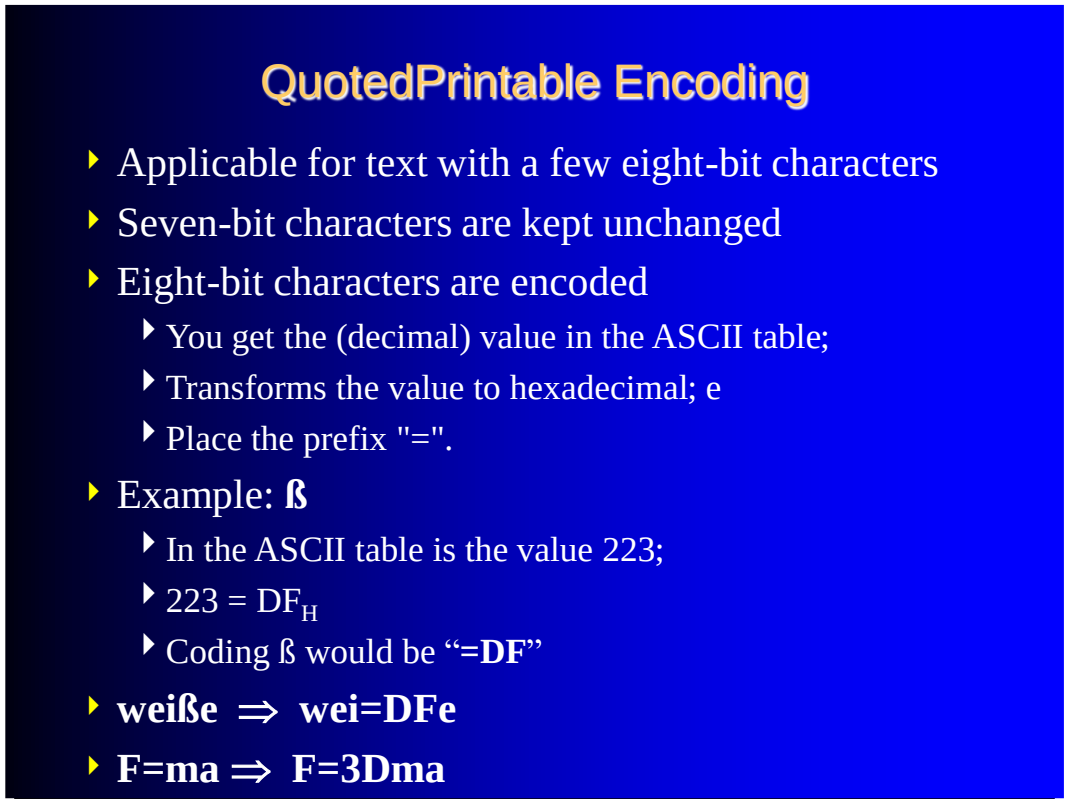
It turns out that three bytes (twenty-four bits) can be transformed into four six-bit values (and vice versa). With this in mind, one can subdivide the entire string of characters into groups of three characters, and transform each crack using the same procedure. For instance:



The input triplet (255, 170, 0) was converted to six-bit sets, with the following values: 111111 (63), 111010 (58), 101000 (40) and 000000 (0), which according to the Base-64 encoding table are the characters /, 6, o and A. These characters will be transmitted in the message and will be decoded in the reverse process by the recipient's reader.

If the size in bytes of the file to be transmitted is exactly divisible by three, the conversion is done without major problems. However, if in the end there are one or two bytes left, the trick is to fill with zeros at the end to complete a triplet. But one must have some way of identifying this to the recipient, otherwise one would not know if they are really zeros or filler. The appropriate solution was to use a 65th character (=) representing that are "artificial" zeros added in the encoding.

To illustrate, the simple byte 255 would be aggregated two zeros (255,0,0). That transformed to six bits would give (63, 48, 0, 0), that would map (/,w,=,=) to be transmitted.



QuotedPrintable

QuotedPrintable works on the premise that most texts use 7-bit characters. To avoid overheading, as well as to maintain readability, 7-bit characters are not encoded.

8-bit characters are converted to a 7-bit (printable character) format as follows::

- Gets the decimal value in the ASCII table;

- The decimal value is encoded in hexadecimal; and

- Place the = character before the hexadecimal value.

For instance:

Paulo Pereira Pinto, pobre pintor português

pinta perfeitamente portas, paredes e pias,

por pouco preço, patrão.

Paulo Pereira Pinto, pobre pintor portugu=EAs

pinta perfeitamente portas, paredes e pias,

por pouco pre=E7o, patr=E3o.

When you want the "=" character to be retained, you must put its hexadecimal value (=3D). So the formula F=ma, encoded using QuotedPrintable, would be: F=3Dma.