Network Working Group
Request for Comments: 2157
Category: Standards Track

H. Alvestrand
UNINETT

January 1998

# Mapping between X.400 and RFC-822/MIME Message Bodies

## Status of this Memo



## Copyright Notice



## Table of Contents

## 1. Introduction

This document is a companion to [MIXER], which defines the principles and translation of headers for interworking between MIME-based RFC-822 mail and X.400 mail.

This document defines how to map body parts of X.400 messages into MIME entities and vice versa, including the handling of multipart messages and forwarded messages.

## 1.1. Glossary

The following terms are defined in this document:

Body part
: Part of a message that has a unique type. This term comes from X.400; the corresponding term in MIME (RFC 2046) is limited to use in parts of a multipart message; the term "body" may correspond better.

Content-type
: Type information indicating what the content of a body part actually is. This term comes from MIME; the corresponding X.400 term is "body part type".

Mapping
: (noun): A description of how to transform an X.400 body part into a MIME body part, or how to transform a MIME body part into an X.400 body part.

Equivalence
: A set of two mappings that taken together provide a lossless conversion between an X.400 body part and a MIME body part

Encapsulation
: The process of wrapping something from one of the mail systems in such a way that it can be carried inside the other mail system. When encapsulating, it is not expected that the other mail system can make reasonable sense of the body part, but a gateway back into the first system will always be able to convert the body part without loss back to its original format.

HARPOON encapsulation
: The encapsulating of a MIME body part by putting it inside an IA5 body with all headers and encoding intact. First described in RFC 1496 [HARPOON].

Tunneling
: What happens when one gateway encapsulates a message and sends it to another gateway that decapsulates it. The hope is that this will cause minimal damage to the message in transit.

DISCUSSION
: At many points in this document, the author has found it useful to include material that explains part of the reasoning behind the specification. These sections all start with DISCUSSION: and continue to the next numbered section heading; they do not dictate any additional requirements on a gateway.

The words MUST, SHOULD and MAY, when capitalized, are used as defined in RFC 2119 [MUST].

## 2. Basic rules for body part conversion

The basic approach for translating body parts is described in section 2.1 and 2.2.

Chapter 3 gives details on "encapsulation", which allows you to be certain that no information is lost even when unknown types are encountered.

Chapter 6 gives the core mappings for various body parts.

The conformance requirements in chapter 8 describe what the minimum conformance for a MIXER gateway is with respect to body part conversion.

DISCUSSION:

At the moment both the MIME and the X.400 worlds seem to be in a stable state of flux with regards to carrying around stuff that is not text. In such a situation, there is little chance of defining a mapping between them that is the best for all people, all of the time. For this reason, this specification allows a gateway considerable latitude in deciding exactly what conversion to apply.

The decision taken by the gateway may be based on various information sources:

(1) If the gateway knows what body parts or content types the recipient is able to handle, or has registered a particular set of preferences for a user, and knows how to convert the message reasonably to those body parts, the gateway may choose to convert body parts in the message to those types only.

(2) If the gateway gets indications (via special headers or heading-extensions defined for the purpose) that the sender wanted a particular representation on the "other side", and the gateway is able to satisfy the request, it may do so. Such a mechanism is defined in chapter 4 of this document.

(3) If the gateway gets a message that might be appropriate to send as one out of several types, but where the typing information does not tell you which one to use (like an X.400 BP14, FTAM "just a file", or MIME application/octet-stream), it may apply heuristics like looking at content or looking at filenames to figure out how to deal with the message.

(4) If the gateway knows that the next hop for the message has limited capabilities (like X.400/84), it may choose to perform conversions appropriate for that medium.

(5) Where no mapping is known by the gateway, it may choose to drop the body part, reject the message, or encapsulate the body part as described in chapter 3. The choice may be configurable, but a conformant MIXER gateway MUST be able to be configured for encapsulation.

In many cases, a message that goes SMTP->X.400->SMTP will arrive without loss of information.

In some cases, the reverse translation may not be possible, or two gateways may choose to apply different translations, based on the criteria above, leading to an apparently inconsistent service.

In addition, service will vary because some gateways will have implemented conversions not implemented by other gateways.

This is believed to be unavoidable.

## 2.1. Generating the IPM Body from MIME

When converting the body of a message from MIME to X.400, the following steps are taken:

If the header does not contain a 822.MIME-Version field, then generate an IPMS.Body with a single IPMS.BodyPart of type IPMS.IA5TextBodyPart containing the body of the RFC 822 message with IPMS.IA5TextBodyPart.parameters.repertoire set to the default (IA5).

If 822.MIME-Version is present, the body is analyzed as a MIME message and the body is converted according to the mappings configured and implemented in the gateway.

## 2.2. Generating the MIME Body from the IPMS.Body

When converting the body of a message from X.400 to MIME, the following steps are taken:

If there is more than one body part, and the first body part is IA5 and starts with the string "RFC-822-Headers:" as the first line, then the remainder of this body part shall be appended to the RFC 822 header. This relies upon the theory that this body part has been generated according to Appendix B of MIXER. A gateway shall check the consistency and syntax of this body part, to ensure that the resulting message is conformant with RFC 822.

If the remaining IPMS.Body consists of a single IPMS.Bodypart, there are three possibilities.

(1) If it is of type IPMS.IA5Text, and the first line is "MIME-Version: 1.0", it is assumed to be a HARPOON-encapsulated body part. The complete body content is then appended to the headers; the separating blank line is inside the message. If an RFC 822 syntax error is discovered inside the message, it may be mapped directly as described below instead.

(2) If it is of type IPMS.IA5Text, then this is mapped directly and the default MIME encoding (7bit) is used, unless very long lines or non-ASCII or control characters are found in the body part, in which case Quoted-Printable SHOULD be used.

(3) All other types are mapped according to the mappings configured and implemented in the gateway.

If the IPMS.Body contains multiple IPMS.Bodypart fields, then a MIME message of content type multipart is generated. If all of the body parts are messages, then this is multipart/digest. Otherwise it is multipart/mixed. The components of the multipart are generated in the same order as in the IPMS.Body.

Each component is mapped according to the mappings configured and implemented in the gateway; any IA5 body parts are checked to see if they are HARPOON mappings, as described above.

## 2.3. Mapping the EMA FTBP parameters

DISCUSSION:

EMA has defined a profile for use of the File Transfer Body Part (FTBP). [MAWG]

New mappings are expected to use this as the mechanism for carrying body parts, and since it is important to have a consistent mapping for the special FTBP parameters, these are defined here.

The mapping of the body will depend on the content-type in MIME and on the application-reference in FTBP, and is not specified here.

However, in many cases, we expect that the translation will involve simply copying the octets from one format to the other; that is, "no conversion".

### 2.3.1. Mapping GraphicStrings

Some parameters of the EMA Profile are encoded as ASN.1 GraphicStrings, which are troublesome because they can contain any ISO registered graphic character set. To map these to ASCII for use in mail headers, the gateway may either:

(1) Use the RFC 2047 [MIME-HDR] encoding mechanism to create appropriate encoded-words for the headers involved. Note that in some cases, such as within Content-Disposition filenames, the encoded-words must be in quotes, which is not the normal usage of encoded-words.

(2) Apply the normalization procedure given in Appendix A to identify the ASCII characters of the string, and replace all non-ASCII characters with the question mark (?).

Both procedures are valid for MIXER gateways; the simplified procedure of ignoring escape sequences and bit-stripping the result is NOT valid.

### 2.3.2. Mapping specific parameters

The following parameters are mapped in both directions:

Content-ID

The mapping of this element is complex.

The Content-ID is encoded as an IPM.MessageIdentifier and entered into the FTBP.FileTransferParameters.related-stored-file. file-identifier.cross-reference.message-reference.

FTBP.FileTransferParameters.related-stored-file. relationship.descriptive-relationship is set to the string "Internet MIME Body Part".

FTBP.FileTransferParameters.related-stored-file. file-identifier.cross-reference.application-crossreference is set to a null OCTET STRING.

The reverse mapping is only performed if the FTBP.FileTransferParameters.related-stored-file. relationship.descriptive-relationship has the string value "Internet MIME Body Part".

Content-Description

The value of this field is mapped to and from the first string in FTBP.FileTransferParameters.environment.user-visible-string.

Content-Disposition

This field is defined in [CDISP]. It has multiple components; the handling of each component is given below.

The "disposition" component is ignored on MIME -> X.400 mapping, and is always "attachment" on X.400 -> MIME mapping.

C-D: filename

The filename component of the C-D header is mapped to and from FileTransferParameters.file-attributes.pathname.

The EBNF.disposition-type is ignored when creating the FTBP pathname, and always set to "attachment" when creating the Content-Disposition header. For example:

Content-Disposition: attachment; filename=dodo.doc

or

Content-Disposition: attachment; filename=/etc/passwd

The filename will be carried as a single incomplete-pathname string. No special significance is assumed for the characters "/" and "\". Note that normal security precautions MUST be taken in using a filename on a local file system; this should be obvious from the second example.

This is done to be conformant with the EMA Profile.

C-D: Creation-date

Mapped to and from FileTransferParameters.file-attributes.date-and-time-of-creation

For this and all other date fields, the RFC-822 date format is used (822.date-time). Note that the parameter syntax of [CDISP] requires that all dates be quoted!

C-D: Modification-date

Mapped to and from FileTransferParameters.file-attributes.date-and-time-of-last-modification

C-D: Read-date

Mapped to and from FileTransferParameters.file-attributes.date-and-time-of-last-read-access

C-D: Size

Mapped to and from FileTransferParameters.file-attributes.object-size. If the value is "no-value-available", the component is NOT generated.

Other RFC-822 headers

Mapped to extension in FTBP.FileTransferParameters.extensions using the rfc-822-field HEADING-EXTENSION from [MIXER].

NOTE:
The set of headers that are mapped will depend on the placement of the body part (single body part or multipart).
When it is the only body of a message, headers starting with "content-" SHOULD be put into the FTAM extension, and all other headers should be put into the IPMS extension for the message.
When it is a single bodypart of a multipart, ALL headers on the body part are included, since there is nowhere else to put them. Note that only headers that start with "content-" have defined semantics in this case.

EMA NOTE

The EMA profile, version 1.5, specifies that handling of extensions is Optional for reception. This means that some non-MIXER gateways may not implement handling of this field, and some UAs may not have the possibility of showing the content of this field to the user.

An alternative approach using FTBP.FileTransferParameters.environment.user-visible-string was suggested to EMA, and the EMA MAWG recommended in its April 1996 conference that the IETF MIXER group should rather choose this approach.

### 2.3.3. Summary of FTBP elements generated

This is a summary of the preceding section, and does not add new information.

The following elements of the FTBP parameters are mapped or used (the rightmost column gives their status in the EMA profile; M=Mandatory, O=Optional, R=Recommended for Origination/Receipt):

| | | |
|---|---|---|
| FileTransferParameters | | M/M |
| Related-Stored-File | | O/O |
| file-identifier | | |
| cross-reference | | |
| application-crossreference | NULL | |
| message-reference | Content-ID | |
| descriptive-relationship | Used as marker | |
| contents-type | Must be unstructured-binary | M/M |
| environment | | M/M |
| application-reference | Selects mapping | M/M |
| user-visible-string | Content-description | R/M |
| file-attributes | | |
| pathname | C-D: Filename | R/M |
| date-and-time-of-creation | C-D: Creation-Date | O/O |
| date-and-time-of-last-modification | C-D: Modification-Date | R/M |
| date-and-time-of-last-read-access | C-D: Read-Date | O/O |
| object-size | C-D: Size | R/M |
| extensions | Other headers | O/O |

All other elements of the FTBP parameters are discarded.

NOTE: There is ongoing work on defining a more complete mapping between FTBP headers and a set of RFC-822 headers. A gateway MAY choose to support the larger set once it is available, but MUST support this limited set.

### 2.4. Information that is lost when mapping

MIME defines fields which add information to MIME contents. Two of these are "Content-ID", and "Content-Description", which have special rules here, but MIME allows new fields to be defined at any time.

The possibilities are limited about what one can do with this information:

(1) When using encapsulation, the information can be preserved

(2) When using mapping to FTBP, the information can be preserved in the FileTransferParameters.extensions defined for that purpose.

(3) When mapping to a single-body message, the information can be preserved as P22 header extensions

(4) When mapping to other body part types, the information must be discarded.

## 3. Encapsulation of body parts

Where no mapping is possible, the gateway may choose any of the following alternatives:

- Discard the body part, leaving a "marker" saying what happened

- Reject the message

- "Encapsulate" the body part, by wrapping it in a body part defined for that purpose in the other mail system

The choice to be made should be configurable in the gateway, and may depend on both policy and knowledge of the recipient’s capabilities.

### 3.1. Encapsulation of MIME in X.400

Four body parts are defined here to encapsulate MIME body parts in X.400.

This externally-defined body part is backwards compatible with RFC 1494. The FTBP body part is compatible with the EMA MAWG document [MAWG], version 1.5, but has some extensions, in particular the one for extra headers.

The imagined scenarios for each body part are:

FTBP For use when sending to recipients that can handle generic FTBP, and for tunnelling MIME to a MIME UA

BP15 For use when tunnelling MIME to a MIME UA through an X.400(88) network, or to UAs that have been written to RFC 1494

IA5 For use when tunneling MIME to a MIME UA through an X.400 network, where some of the links may involve X.400(84).

BP14 For use when the recipient may be an X.400(84) UA with BP14 handling capability, and the loss of information in headers is not regarded as important.

but the gateway is free to use any method it finds appropriate in any situation.

FTBP is expected to be the most useful body part in sending to X.400(92) systems, while the BP14 content passing is primarily useful for sending to X.400(84) systems.

#### 3.1.1. FTBP encapsulating body part

This body part utilizes the fundamental assumption in MIME that all message content can be legally and completely represented by a single octet stream, the "canonical format".

The FTBP encapsulating body part is defined by the application-reference id-mime-ftbp-data; all headers are mapped to the FTBP headers, including putting the "Content-type:" header inside the FTBP ExtensionsField.

Translation from the MIME body part is done by:

- Undoing the content-transfer-encoding

- Setting the "FileTransferData.FTdata.value.octet-aligned" to the resulting string of octets

- Putting the appropriate parameters into the headers.

Reversing the translation is done by:

- Extracting the headers

- Applying an appropriate content-transfer-encoding to the body. If this is for some reason different from the content-transfer-encoding: header retrieved from the headers, the old one must be deleted.

This mapping is lossless, and therefore counts as "no conversion".

Note that this mapping does not work with multipart types; the multipart must first be mapped to a ForwardedIPMessage.

### 3.1.2. BP15 encapsulating body part

This section defines an extended body part, based on body part 15, which may be used to hold any MIME content.

```
 mime-body-part EXTENDED-BODY-PART-TYPE
       PARAMETERS MimeParameters
                IDENTIFIED BY id-mime-bp-parameters
       DATA            OCTET STRING
       ::= id-mime-bp-data

 MimeParameters ::=
       SEQUENCE {
                  content-type       IA5String,
                  content-parameters SEQUENCE OF
                                     SEQUENCE {
                                         parameter          IA5String
                                         parameter-value    IA5String
                                     }
                  other-header-fields RFC822FieldList
              }
```

The OBJECT IDENTIFIERS id-mime-bp-parameter and id-mime-bp-data are defined in Appendix B. A MIME content is mapped onto this body part. The MIME headers of the body part are mapped as follows:

RFC822FieldList is defined in Appendix L of [MIXER].

Content-Type:
The "type/subtype" string is mapped to MimeParameters.content-type.

For each "parameter=value" string create a MimeParameters.content-parameters element. The MimeParameters.content-Parameters.parameter field is set to the parameter and the MimeParameters.content-parameters.parameter-value field is set to the value.

Quoting is preserved in the parameter-value.

Other
Take all other headers and create MimeParameters.other-header-fields.
The MIME-version, content-type and content-transfer-encoding fields are NOT copied.

NOTE:
The set of headers that are mapped will depend on the placement of the body part (single body part or multipart).
When it is the only body of a message, headers starting with "content-" SHOULD be put into the other-header-fields, and all other headers should be put into the IPMS extension for the message.
When it is a single bodypart of a multipart, ALL headers on the body part are included, since there is nowhere else to put them. Note that only headers that start with "content-" have defined semantics in this case.

The body is mapped as follows:

Convert the MIME body part into its canonical form, as specified in Appendix H of MIME [MIME]. This canonical form is used to generate the mime-body-part.data octet string.

The Parameter mapping may be used independently of the body part mapping (e.g., in order to use a different encoding for a mapped MIME body part).

This body part contains all of the MIME information, and so can be mapped back to MIME without loss of information.

The OID id-mime-bp-data is added to the Encoded Information Types of the envelope.

This body part is completely compatible with RFC 1494.

When converting back to a MIME body part, the gateway is responsible for:

(1) Selecting an appropriate content-transfer-encoding, and deleting any content-transfer-encoding header from the other-header-fields

(2) Adding quotes to any parameters that need them (but not adding quotes to parameters that are already quoted)

(3) Removing any content-type field that is left in the RFC822FieldList of the message that is redundant or conflicting with the one from the mime-body-part

(4) Make sure that on multipart messages, the boundary string actually used is reflected in the boundary-parameter of the content-type header, and does not occur within the body of the message.

### 3.1.3. Encapsulation using IA5 (HARPOON)

This approach is the one taken in RFC 1496 - HARPOON - for tunneling any MIME body part through X.400/84 networks. It has proven rather unhelpful for bringing information to X.400 users, but preserves all the information of a MIME body part.

The following IA5Text body part is made:

- Content = IA5String

- First bytes of content: (the description is in US ASCII, with C escape sequences used to represent control characters):

```
MIME-version: <version>\r\n
Content-type: <the proper MIME content type>\r\n
Content-transfer-encoding: <7bit, quoted-printable or base64>\r\n
<Possibly other Content headings here, terminated by\r\n>
 \r\n
<Here follows the bytes of the content, encoded
 in the proper encoding>
```

All implementations MUST place the MIME-version: header first in the body part. Headers that are placed by [MIXER] into other parts of the message MUST NOT be placed in the MIME body part.

This encapsulation may also be applied to subtypes of multipart, creating a single IA5 body part that contains a single multipart/*, which in turn may contain multiple MIME body parts.

### 3.1.4. Content passing using BP14

This is described in this section because it is at the same conceptual level as encapsulation. It is a lossy transformation; it is impossible to reconstruct the MIME type information from it.

Nevertheless, there is a demand for such functionality.

This "encapsulation" simply strips off all headers, undoes the content-transfer-encoding, and creates a BilaterallyDefined body part (BP14) from the resulting octet stream.

No reverse translation is defined; when a BP14 arrives at a MIXER gateway, it will be turned into an application/octet-stream according to chap. 6.3

## 3.2. Encapsulating X.400 Body Parts in MIME

This section specifies a generic mechanism to map X.400 body parts to a MIME content. This allows for the body part to be tunneled through MIME. It may also be used directly by an appropriately configured MIME UA.

This content-type is defined to carry any X.400 extended body part. The mapping of all standard X.400 body parts is defined in this document. The content-type field is "application/x400-bp". The parameter is defined by the EBNF:

```
    mime-parameter =  "bp-type=" ( object-identifier / 1*DIGIT=
```

If the body is a basic body part, the bp-type parameter is set to the number of the body part’s context-specific tag, that is, the tag of the IPMS.Body.BodyPart component.

If the body is an Extended Body Part, the EBNF.object-dentifier is set to the OBJECT IDENTIFIER from IPMS.body.externally-defined.data.direct-reference.

For example, a basic VideotexBodyPart will have

```
   Content-type=application/x400-bp; bp-type=6
```

whilst a Extended Videotex body part will have

Content-type=application/x400-bp; bp-type=2.6.1.4.5

The body contains the raw ASN.1 IPM body octet stream, that is, the BER encoding of the IPM.Body.BodyPart, including the initial tag octet. The content may use a content-transfer-encoding of either base64 or quoted-printable when carried in 7-bit MIME. It is recommended to use the one which gives the more compact encoding of the data. If this cannot be determined, Base64 is recommended. No attempt is made to turn the parameters of Extended Body Parts into MIME parameters, as this cannot be done in a general manner.

For extended body parts, the

## 3.3. Encapsulating FTBP body parts in MIME

The File Transfer Body Part is believed to be important in the future as "the" means of carrying well-identified data in X.400 networks.

They also share the property (at lest when limited to the EMA MAWG functional profile) of having a well-defined data part that is always representable as a sequence of bytes.

This conversion will have to fail, and the x400-bp encapsulation used instead, if:

- FileTransferData has more than one element
- Contents-type is not unstructured-binary
- Parameters that are not mappable, but important, are present (like Compression, which EMA doesn’t recommend).

Otherwise, it can be encapsulated in MIME by:

- Creating the "content-type" value by forming the string "application/x-ftbp." and appending the numbers of the OID found in FileTransferParameters.environment.application-reference.registered-identifier
- Mapping all other parameters according to the standard FTBP parameter mapping
- Applying an appropriate content-transfer-encoding to the data contained in FileTransferData.value.encoding

DISCUSSION:

The choice of the somewhat strange, and by necessity unregistered, MIME type "application/x-ftbp.n.n.n.n" is because for any concrete example of this usage, it will be easy to configure any MIME reader to take advantage of the identification. If the MIME type registration rules are ever changed to allow the registration of a namespace, rather than just of names, the "x-" can be deleted, and the types can be "application/ftbp.n.n.n.n".

## 4. User control over the gateway choice

In some cases, the gateway may make an inappropriate choice when deciding what to do about a particular body part.

To allow an escape clause, this chapter defines a way in which the user can signal the gateway what action it finds most appropriate.

The headers given here override any "conversion prohibited" and "conversion with loss prohibited" on the message.

It is still the gateway's responsibility that the generated messages conform to the destination domain's syntax rules.

DISCUSSION:

The intent of this mechanism is to allow the sender to efficiently get a message through to a single recipient when the sender has information about the recipient that the gateway does not have.

It is not a part of the minimum functionality listed in chapter 8; a gateway does not have to implement this spec to be MIXER conformant, but if implemented, it should be done like this.

The additional complexity, both in user interface and in protocol, of making this field selectable per recipient was not thought worthwhile;

### 4.1. Conversion from MIME to X.400

The header field described below specifies explicit MIXER conversion. Comments are allowed within the field according to the usual RFC 822 convention.

If "x400-object-id" is omitted, "tunnel" is assumed.

```
mime-to-x400 = "Wanted-X400-Conversion" ":"
                [ mime-from ]  [ x400-object-id ]
```

"in" x400-encoding

```
x400-object-id =  "to" ( object-identifier-2 / "tunnel" )
x400-encoding = "bp14" / "bp15" / "ftbp" / "ia5"
mime-from = "from" mime-type
mime-type = word
```

There is no way to ask for a different conversion based on MIME parameters or bodypart content.

Examples:

```
Wanted-X400-Conversion: from application/msword
                to 1.2.840.113556.4.2 (Microsoft defined ms-word)
                in ftbp
```

This uses the MAWG definitions, and leads to an FTBP encoding.

```
Wanted-X400-Conversion: from application/msword
               to tunnel in bp14
```

This leads to a Body Part 14 encoding for all body parts of type application/msword.

```
Wanted-X400-Conversion: in bp14
```

This requests that this specific body part be encoded in Body Part 14.

This field may be used in two places:

(1) In the heading of an unstructured MIME body part. In this case the EBNF.mime-from is omitted, and the requested conversion applies to the body part.

(2) In a multipart. In this case, the body part type to which the conversion applies is defined by EBNF.mime-from, and the conversion applies to all body parts of this MIME type contained in the multipart, including those contained in nested messages and multiparts. If a contained body part has its own heading, this takes precedence. Note that the "from" parameter is mandatory when used in a multipart.

The EBNF.x400-object-id shall be present when "bp15" or "ftbp" encoding is selected.

The value "tunnel" implies encapsulation as defined in Chapter 3.

The "object identifier" used below is:

- For BP 15, it is the value of the EXTENDED-BODY-PART-TYPE macro that defines the body part, which is found in ExternallyDefinedBodyPart.data.direct-reference.

- For FTBP, it is the value of the Environment.application-reference.

## 4.2. Conversion from X.400 to MIME

The IPM heading defined here shall be present in the heading of a message. It defines the mapping for all body parts of the specified types, including those in nested messages.

```
wanted-MIME-conversion HEADING-EXTENSION
        VALUE WantedMIMEConversions
        ::= id-wanted-MIME-conversions


WantedMIMEConversions ::= SEQUENCE OF X400toMIMEConversion

X400toMIMEConversion ::= SEQUENCE {
        x400-type X400Type,
        mime-type MIMEType }


X400Type ::= CHOICE {
        standard [0] INTEGER,            -- standard body part
        extended [1] OBJECT IDENTIFIER,  -- BP 15
        ftbp     [2] OBJECT IDENTIFIER}     -- FTBP
                                            -- application-reference

MIMEType ::= SEQUENCE {
        type IA5String,         -- type (e.g., application/ms-word)
        encoding [1] IA5String OPTIONAl -- e.g. quoted-printable
        parameters [2] IA5String OPTIONAL }     -- MIME Parameters
```

The heading extension includes all requested conversions, with explicit information as to how each body part type is encoded in MIME.

FTBP is identified as a separate body part type, as there will be a need for different encodings, dependent on what is being carried.

Encapsulation is requested by asking for "application/x400-bp" or "application/ftbp" as the destination type.

For FTAM body parts, the parameters will survive the gatewaying process. For other body parts, there are three alternatives:

(1) The gateway knows a defined mapping for this particular body part and destination type. It will be used, and parameters mapped accordingly.

(2) The gateway knows how to extract an OCTET STRING from the body part, and the destination is a simple MIME body part. All information outside the OCTET STRING is lost. (This may be the case for a BP14 that should end up in an application/xyzzy, for instance).

(3) The gateway knows of no relevant mapping, and does not know how to simplify the X.400 body part. The gateway will then proceed as if the mapping control field had not been present.

## 5. The equivalence registry

### 5.1. What information one must give about a mapping

The following information MUST be supplied when describing an equivalence or a mapping:

MIME type name (which must be preregistered)

X.400 body part (often BP15 or FTAM Body Part)

If BP15 is used, the following information must be given:

(1) Object Identifier for X.400 BP15 Data

(2) Object Identifier for X.400 BP15 Parameters

(3) X.400 ASN.1 Syntax (must be an EXTENDED-BODY-PART-TYPE macro)

If FTBP is used, the following information must be given:

<1) Object Identifier for the FTAM Environment.application-reference

<2) Object Identifier for the FTAM Contents-type, if unstructured-binary is not used

(3) Any other special considerations

In all cases, the following must be given:

Conversion algorithms. The expected effect of "Conversion prohibited" and "Conversion with loss prohibited" should be noted.

The conversion must be specified with enough detail to permit independent implementation; literature references are acceptable.

An equivalence can be registered with IANA using the form at the end of this document. The purpose of the registration is to achieve a greater uniformity among gateways implementing the same translation; there is no requirement that a gateway must support all of the translations that are registered with IANA, and there is no requirement that all conversions supported by a gateway are registered with IANA. Specific conformance requirements for MIXER are given at the end of this document.

Anyone can register an equivalence with IANA, and may update the registered equivalence at any time, or reassign the right to update the registry entry at any time. However, the IESG has the power to "lock" a registration, so that changing it requires IESG approval, and to update such a "locked" registration. All registered equivalences defined in standards-track documents (including this one) are locked.

## 5.2. Equivalence summary for known X.400 and MIME Types

This section itemizes the equivalences for all currently known MIME content-types and X.400 body parts.

For each MIME content-type/X.400 body part pair, the equivalence table contains an entry with the following sections:

X.400 Body Part
: This section identifies the X.400 Body Part governed by this Table entry. It includes any OBJECT IDENTIFIERs or other parameters necessary to uniquely identify the Body Part.

MIME Content-Type
: This section identifies the MIME content-type governed by this Table entry. The MIME content-type named here must be registered with the IANA.

Section/document reference
: Reference to section of this document, or to the other document that describes this mapping.

The initial Equivalence Table entries in this document are described using this convention.

Further registrations of equivalences should be submitted to the IANA after a public review, using the example form given at the end of this document.

### 5.3. MIME to X.400 Table

| MIME content-type | X.400 Body Part | Section |
|---|---|---|
| text/plain | | |
| charset=us-ascii | ia5-text | 6.1 |
| charset=ISO-8859-x | EBP - GeneralText | 6.2 |
| text/richtext | no mapping defined | Encap |
| application/oda | EBP - ODA | [ODA] |
| application/octet-stream | bilaterally-defined or | 6.3 |
| | FTBP unknown attachment | 6.4 |
| application/postscript | EBP - mime-postscript-body | [POSTSCRIPT] |
| image/g3fax | g3-facsimile | [IMAGES] |
| image/jpeg | EBP - mime-jpeg-body | [IMAGES] |
| image/gif | EBP - mime-gif-body | [IMAGES] |
| audio/basic | no mapping defined | Encap |
| video/mpeg | no mapping defined | Encap |
| message/RFC822 | ForwardedIPMessage | 6.5 |
| multipart/* | ForwardedIPMessage | 6.6 |
| multipart/signed | HARPOON encap | 7.3 |
| multipart/encrypted | HARPOON encap | 7.4 |

Abbreviation: EBP - Extended Body Part

### 5.4. X.400 to MIME Table

Basic Body Parts

| X.400 Basic Body Part | MIME content-type | Section |
|---|---|---|
| ia5-text | text/plain;charset=us-ascii | 6.1 |
| voice | No Mapping Defined | Encap |
| g3-facsimile | image/g3fax | [IMAGES] |
| g4-class1 | no mapping defined | Encap |
| teletex | text/plain;charset=teletex | 6.7 |
| videotex | no mapping defined | Encap |
| encrypted | no mapping defined | Encap |
| bilaterally-defined | application/octet-stream | 6.3 |
| nationally-defined | no mapping defined | Encap |
| externally-defined | See Extended Body Parts below | |

| | | |
|---|---|---|
| ForwardedIPMessage | message/RFC822 or multipart | 6.5,6.6 |

| X.400 Extended Body Part | MIME content-type | Section |
|---|---|---|
| GeneralText | text/plain;charset=ISO-8859-x | 6.2 |
| ODA | application/oda | [ODA] |
| mime-postscript-body | application/postscript | [POSTSCRIPT] |
| mime-jpeg-body | image/jpeg | [IMAGES] |
| mime-gif-body | image/gif | [IMAGES] |
| FTAM | various | 2.3,6.4 |

| FTAM application ID | MIME content type | Section |
|---|---|---|
| ema-unknown-attachment | application/octet-stream | 6.4 |

## 5.5. Use of OBJECT IDENTIFIERs and ASN.1 MACROS

When one wants to define new BP15 body parts for use with equivalences, it is important to know that X.420 dictates that Extended Body Parts shall:

(1) use OBJECT IDENTIFIERs (OIDs) to uniquely identify the contents, and

(2) be defined by using the ASN.1 Macro:

```
EXTENDED-BODY-PART-TYPE MACRO::=
BEGIN
   TYPE NOTATION  ::= Parameters Data
   VALUE NOTATION ::= value (VALUE OBJECT IDENTIFIER)

   Parameters     ::=  "PARAMETERS" type "IDENTIFIED"
                       "BY" value(OBJECT IDENTIFIER)
                     | empty;
   Data           ::= "DATA" type
END
```

To meet these requirements, this document uses the OID

```
mixer
```

defined in [MIXER], as the root OID for X.400 Extended Body Parts defined for MIME interworking.

Each Extended Body Part contains Data and optional Parameters, each being named by an OID. To this end, two OID subtrees are defined under mixer-bodies, one for Data, and the other for Parameters:

```
   mixer-bp-data  OBJECT IDENTIFIER ::=
                  { mixer 1 }

   mixer-bp-parameter OBJECT IDENTIFIER ::=
                  { mixer 2 }
```

All definitions of extended X.400 body parts submitted to the IANA for registration with a mapping must use the Extended Body Part Type macro for the definition. See [IMAGES] for an example.

Lastly, the IANA will use the mixer-bp-data and mixer-bp-parameter OIDs as root OIDs for any new MIME content-type/subtypes that aren't otherwise registered in the Equivalence Table.

NOTE: The ASN.1 for an ExternallyDefinedBodyPart is

```
   ExternallyDefinedBodyPart ::= SEQUENCE {
     parameters [0] ExternallyDefinedParameters OPTIONAL,
     data           ExternallyDefinedData }

   ExternallyDefinedParameters ::= EXTERNAL

   ExternallyDefinedData ::= EXTERNAL
```

The ASN.1 for EXTERNAL is (from X.208):

```
   EXTERNAL ::= [UNIVERSAL 8] IMPLICIT SEQUENCE
   {direct-reference     OBJECT IDENTIFIER OPTIONAL,
   indirect-reference    INTEGER OPTIONAL,
   data-value-descriptor ObjectDescriptor OPTIONAL,
   encoding CHOICE
     {single-ASN1-type  [0] ANY,
      octet-aligned     [1] IMPLICIT OCTET STRING,
      arbitrary         [2] IMPLICIT BIT STRING}}

   ObjectDescriptor ::= [UNIVERSAL 7] IMPLICIT GraphicString
```

There are a bit too many choices here; the common X.400 usage for BP15 encoding is to:

(1) Always use direct-reference

(2) Omit indirect-reference and data-value-descriptor

(3) Use the single-ASN1-type encoding only

Unfortunately, some implementations have chosen to use the octet-aligned choice when constructing values where the ASN.1 type is OCTET STRING, which of course caused interoperability problems.

An attempt to specify that X.420 only allowed the single-ASN1-type choice in the 1996 versions is still (Sept 1995) being debated in ISO; the end result seems to be that all agree in principle that single-ASN1-type should be used, but that one has to allow the generation of the octet-aligned choice as being conformant.

## 6. Defined Equivalences

### 6.1. IA5Text - text/plain

X.400 Body Part: IA5Text MIME Content-type: text/plain; charset=US-ASCII Conversion Type: No conversion Comments:

When mapping from X.400 to MIME, the "repertoire" parameter is ignored.

When mapping from MIME to X.400, the "repertoire" parameter is set to IA5 (5).

NOTE: The MIME Content-type headers are omitted, when mapping from X.400 to MIME, if and only if the IA5Text body part is the only body part in the IPMS.Body sequence.

NOTE: IA5Text specifies the "currency" symbol in position 2/4. This is converted without comment to the "dollar" symbol, since the author of this document has seen many documents in which the position was intended to indicate "dollar" while he has not yet seen one in which the "currency" symbol is intended.

(For reference: The T.50 (1988) recommendation, which defines IA5, talks about ISO registered set number 2, while ASCII, using the "dollar" symbol, is ISO registered set number 6. There are no other differences.)

NOTE: It is not uncommon, though it is a violation of the standard, to use 8-bit character sets inside an IA5 body part. Gateways that can expect to encounter this situation should consider implementing something like the guidance given in RFC 1428 [MIMETRANS], "Transition of Internet Mail from just-send-8 to 8-bit SMTP/MIME", and generate appropriate charset parameters for the MIME messages they generate. This behavior is not required for MIXER conformance, since it is only needed when the base standards are violated.

### 6.2. GeneralText - text/plain (ISO-8859)

X.400 Body Part: GeneralText; CharacterSets in
6, 14, 42, 87, 100,101,109,110,126,127,138,144,148
MIME Content-Type: text/plain; charset=ISO-8859-(1-9)
or iso-2022-jp
Conversion Type: Text conversion without character change When mapping from X.400 to MIME, the character-set is chosen from the table below according to the value of Parameters.CharacterSets. If no match is found, and the gateway does not support a conversion, the character set shall be encoded as x-iso-nnn-nnn-nnn, where "nnn" is the numbers of the Parameters.CharacterSets, sorted in numeric order.

When mapping from MIME to X.400, GeneralText is an Extended Body Part, hence it requires an OID. The OID for the GeneralText body is defined in [MOTIS], part 8, annex D, as {2 6 1 4 11}. The OID for the parameters is {2 6 1 11 11}.

The Parameters.CharacterSets is set from table below according to the value of "charset"

The following table lists the MIME character sets and the corresponding ISO registry numbers. If no correspondence is found, this conversion fails, and the generic body part approach is used.

| MIME charset | ISO IR numbers | Comment |
|---|---|---|
| ISO-8859-1 | 6, 100 | West European "8-bit ASCII" |
| ISO-8859-2 | 6, 101 | East European |
| ISO-8859-3 | 6, 109 | <regarded as obsolete> |
| ISO-8859-4 | 6, 110 | <regarded as obsolete> |
| ISO-8859-5 | 6, 144 | Cyrillic |
| ISO-8859-6 | 6, 127 | Arabic |
| ISO-8859-7 | 6, 126 | Greek |
| ISO-8859-8 | 6, 138 | Hebrew |
| ISO-8859-9 | 6, 148 | Other Latin-using languages |
| ISO-2022-JP | 6, 14, 42, 87 | Japanese |

When converting from MIME to X.400, generate the correct OIDs for use in the message envelope’s Encoded Information Types by looking up the ISO IR numbers in the above table, and then appending each to the id-cs-eit-authority {1 0 10021 7 1 0} OID, generating 2-4 OIDs.

Similar procedures can be used with other MIME charsets that map to a set of ISO character sets.

The escape sequences to designate and invoke the relevant character sets in their proper positions must be added to the front of the GeneralText character string.

For ISO 8859-1, the relevant escape sequence will be:

ESC 28 42
ASCII in G0

ESC 2D 41
ISO-IR-100 in G1

ESC 21 41
High control character set in C1

ESC 7E
Locking shift 1 Right

These escape sequences are removed when converting from GeneralText to text/plain.

Note that new character sets may be defined on both the Internet side and the X.400 side; a gateway MAY choose to implement more conversions in the same fashion.

DISCUSSION:

The conversion of text is a problematic one, and one in which it is likely that gateways should be given wide latitude to make decisions based upon their knowledge of the user's preferences. The text given below is thought to give the best approximation to a gateway conforming to current and anticipated usage in the MIME and X.400 worlds, and is the way recommended when no knowledge of the recipient's capabilities exists.

The lossless changes, such as normalizing escape sequences, can be done even when "conversion-prohibited" is set. If "conversion-with-loss-prohibited" is set, translation to a character set that is not able to encode all characters cannot be done, and the message should be non-delivered with an appropriate non-delivery reason.

The common use of character sets in MIME is somewhat different from the rules given by X.400; in particular, it is common in MIME to assume that the character sets follow strict rules. For the ISO-8859-x character sets, it is assumed that they are designated and invoked at the beginning of the text, and that no designation or invocation sequences occur within the body of the text.

The rules for ISO-2022-JP are given in RFC 1468 [2022-JP], and are even more particular, using a pure 7-bit encoding in which each line of text starts in ASCII.

Therefore, the text must be "normalized" by going through the whole message, using a state machine or similar device to remove or rewrite all escape and shift sequences.

Appendix A gives pseudocode for such a conversion.

NOTE: In 1988, the GeneralText body part was defined in ISO 10021-8 [MOTIS], and NOT in the corresponding CCITT recommendation; this was added later. Also, the parameters have been heavily modified; they should be a SET OF INTEGER in the currently valid text. Use the latest version of the standard that you can get hold of.

### 6.3. BilaterallyDefined - application/octet-stream

X.400 Body Part: BilaterallyDefined
MIME Content-Type: Application/Octet-Stream (no parameters)
Conversion Type: No conversion

When mapping from MIME to X.400, if there are parameters present in the Content-Type: header field, they are removed.

DISCUSSION:

The parameters "name" "type" and "conversions" are advisory; name and conversions are depreciated in RFC 2046.

The parameter "padding" changes the interpretation of the last byte of the data, but it is deemed better by the WG to delete this information than to non-deliver the body part. The "padding" parameter is rarely used with MIME.

Use of BilaterallyDefined Body Parts is specifically deprecated in both 1988 and 1992 X.400. It is retained solely for backward compatibility with 1984 systems, and because it is in common use.

### 6.4. FTBP EMA Unknown Attachment - application/octet-stream

X.400 Body Part: FTBP EMA Unknown Attachment
MIME Content-Type: Application/Octet-Stream
Conversion Type: No conversion

The OID for the Unknown Attachment is { joint-iso-ccitt(2) country(16) us(840) organization(1) ema(113694) objects(2) messaging(2) attachments(1) unknown(1) }, or 2.16.840.1.113694.2.2.1.1 for short.

NOTE: Previous EMA drafts gave it as { iso(1) countries(2) usa(840) organization (1) ema (113694) objects(2) messaging(2) attachments(1) unknown (1)}, or 1.2.840.1.113694.2.2.1.1 for short.

The parameters for this type must be mapped according to chapter 2.3, with the following extensions for the parameters of the application/octet-stream:

> If there is no Content-Disposition parameter with a filename, and there is a name parameter, the FTBP.FileTransferParameters.File-attributes.pathname is generated from this parameter. Note that RFC 2046 recommends not using the "name" parameter.

The "type", "conversions" and "padding" attributes are ignored; "type" is for human consumption; "conversions" are discouraged in RFC 2046.

The body mapping is just copying the bytes in both directions.

## 6.5. MessageBodyPart - message/RFC822

X.400 body part: MessageBodyPart
MIME Content-Type: message/RFC822
Conversion Type: Special

NOTE: If the headers of the X.400 MessageBodyPart contains the "multipart-message" heading extension with the isAMessage bit set (either explicitly or implicitly), the mapping should be to multipart/* according to section 6.6, below.

To map an IPMS.MessageBodyPart, the full X.400 -> RFC 822 mapping is recursively applied, to generate an RFC 822 Message. If present, the IPMS.MessageBodyPart.parameters.delivery-envelope is used for the MTS Abstract Service Mappings. If present, the IPMS.MessageBodyPart.parameters.delivery-time is mapped to the extended RFC 822 field "Delivery-Date:".

When a message/RFC822 is contained within a MIME message, it is mapped to an IPMS.MessageBodyPart according to MIXER. specification. Any mappings that would have been made to the MTS Abstract Service are placed in IPMS.MessageBodyPart.parameters.delivery-envelope.

### 6.6. MessageBodyPart - multipart/*

X.400 body part: MessageBodyPart
MIME Content-Type: multipart/*
Conversion Type: Special

NOTE: If the headers of the X.400 MessageBodyPart do not contain the "multipart-message" heading extension with the "isAMessage" flag FALSE=, the mapping should be to message/RFC822.

A MIME multipart is a set of content-types and not a message with a set of content types. When the multipart is at the outermost MIME header, elements of the multipart are mapped directly onto IPMS.Bodypart.

When the MIME multipart is not at the outermost level, it is mapped to an IPMS.MessageBodyPart containing an IPMS.Bodypart for each element of the multipart.

When a nested IPMS.Message is generated from a multipart, an IPMS.heading shall always be generated. The only mandatory field is the IPMS.Heading.this-IPM message id, which shall be generated by the gateway. An IPMS.Heading.subject field shall also be generated, in order to provide useful information to non-MIME capable X.400(88) UAs and to all X.400(84) UAs. The subject field is set as follows according to the multipart subtype:

mixed:
"Multipart Message"

alternative:
"Alternative Body Parts containing the same information"

digest:
"Message Digest"

parallel:
"Body Parts interpreted in parallel"

other:
"Multipart Message (<subtype>)"

For other types of multipart, the multipart subtype shall be included in the subject line.

For each multipart, the following IPMS.HeadingExtension shall be generated, with the value set according to the subtype.

If the multipart is the outermost multipart, and the subtype is "mixed", it may be omitted.

```
multipart-message HEADING-EXTENSION
        VALUE MultipartType
         ::= id-hex-multipart-message-v2

MultipartType ::= SEQUENCE {
              subtype IA5String,
              isAMessage BOOLEAN DEFAULT TRUE }
```

The MultipartType contains the subtype, for example "digest". If this heading is present when mapping from X.400 to MIME, the appropriate multipart may be generated.

The isAMessage flag is needed because of the case where a message contains a ForwardedIPMessage, which itself was generated from a MIME message that was a Multipart; it is set whenever the multipart is the outermost level of nesting inside a Message/RFC822.

NOTE:
When downgrading to X.400/84, the content-type SHOULD be regenerated from this heading-extension and put into the RFC-822-HEADERS extra body part.

NOTE:
This definition is different from the one in RFC 1494, because the RFC 1494 definition turned out to be insufficient when new subtypes of Multipart (like Signed or Related) were defined. That is the reason for the "-v2" part of the name of the OID.

If both the old and the new heading extensions occur on a message, a MIXER gateway should give preference to the new one.

### 6.7. Teletex - Text/Plain (Teletex)

X.400 Body Part: Teletex
MIME Content-Type: text/plain; charset=Teletex
Conversion Type: Text conversion

From X.400 to RFC-822, the conversion shall take the bytes of all the pages in the "data" part of the TeletexBodyPart, add a FF character (0x0C, control-L) to each part that does not already end in one, and concatenate them together to form the body of the Text/Plain.

The character set shall be "Teletex", which is especially registered for this purpose. Its definition is shown in an appendix.

The parameters are discarded.

From RFC-822 to X.400, the conversion shall split the content at each occurrence of the FF character (0x0C), delete the character and construct the Teletex body part as a SEQUENCE OF TeletexString, as described in X.420(88), section 7.3.5

The TeletexParameters may, but need not, contain the number-of-pages component.

NOTE: It is recommended, but not mandated, that the data be converted into a more widespread character set like ISO-8859-1 or ISO-2022-JP (if applicable) if possible. This will result in the reverse translation giving a GeneralText body part, which will have to be dealt with appropriately at the X.400/88 to X.400/84 downgrading boundary, if possible, but will give a much greater chance that the MIME recipient can actually read the message.

DISCUSSION:

The Teletex body part is frequently used in X.400(84) to send around text with slightly extended character sets beyond ASCII.

Its body consists of a series of "pages", separated by ASN.1 representation. It is important to many people to have this mapped into something that is readable to most end-users; therefore, it is recommended to map this onto Text/Plain; however, since this is not plain text, the conversion must be specified.

Note that the definition of Text/Plain permits only CRLF as a line separator; the sequences "CR FF" and "CR LF LF LF.." permitted in Teletex must be encoded as Quoted-Printable.

## 7. Body parts where encapsulation is recommended

Some body parts are MIME constructs, and their functionality will be severely damaged if they are coerced into an X.400 framework.

Special care needs to be taken with these; they are described below.

### 7.1. message/external-body

The gateway MUST support the encapsulation of this body part using the HARPOON encapsulation (IA5).

It MAY support some kind of retrieval of the referred object.

DISCUSSION:

The message/external-body part points to an object that can be retrieved using Internet protocols.

There are three cases to consider for the recipient's capabilities:

(1) The user has no Internet access. In this case, the user might be grateful if the gateway fetches the body part and inserts it into the message. If the body part is large or dynamic, it might not be appropriate.

(2) The user has Internet access, but no UA support for fetching external-body objects.

(3) The user has Internet access and UA support for fetching external-body objects, based on an understanding of this document.

Some access-types, like anonymous FTP, are easy to resolve. Others, like the Mailserver access-type, are almost impossible to resolve at a gateway.

To support the second case above, the tunneling method chosen is the HARPOON encapsulation described in section 3.1.3, using an IA5 body part, inserting the string "MIME-Version: 1.0 (generated by gateway)" at the beginning of the body part. (The part in parentheses can be changed at will).

This will:

(1) Maximize the chance that the user will see the message

(2) Give the user hints that will enable him to fetch the message using other Internet tools

(3) Identify the message as a MIME object in a reliable fashion, allowing UAs to support the fetching of the object if the UA implementor desires.

### 7.2. message/partial

This represents part of a larger message, where it is only possible to parse the complete message after getting all the pieces.

The gateway MUST support the encapsulation of this body part.

It MAY implement transparent reassembly of the message, but in this case, it MUST support a configurable timeout for the reassembly, defaulting back to encapsulation.

DISCUSSION:

The gateway's choices are:

(1) Wait until all the pieces arrive at the gateway, reassemble the message, and use normal processing

(2) Encapsulate the message, using any encapsulation method (BP15, FTAM or HARPOON).

In some cases, not all pieces will arrive at the gateway; some may have been transferred through other gateways due to route changes or machine outages; some may have been lost in transit.

### 7.3. multipart/signed

A gateway MUST implement encapsulation of multipart/signed using HARPOON.

The gateway MAY be configured to do other processing, as outlined in the discussion below. This is outside the scope of the standard.

DISCUSSION:

Gatewaying security is a problem. The gateway can basically take three approaches:

- Strip the multipart/signed, leaving the bare body part unsecured, possibly with a comment that the signature was stripped

- Attempt to check the signature and re-signing the message using X.400 security functions, then stripping as above

- Encapsulate the message. This is the only approach that allows end to end security, but requires MIME functionality at the recipient.

- Replace the message content with multiple body parts, containing first an unsecured body part and then the encapsulated multipart/signed.

All these are valid options for a MIXER gateway.

Note that the encapsulation must use HARPOON, as the signature is computed on the ENCODED body part, not on the canonical representation, and HARPOON is the only encapsulation that preserves the content transfer encoding of the message.

Note also that all methods except for encapsulation break end-to-end security; the recipient can place no more trust in the integrity of the message than he can place in the security of the gateway.

### 7.4. multipart/encrypted

A gateway MUST implement encapsulation of multipart/encrypted using HARPOON.

If the implementor chooses to allow other processing at the gateway, as outlined below, he/she is advised that there are grave security concerns with such a solution, since it violates the general rule of keeping decryption keys as close to the user as possible.

DISCUSSION:

There are two basic cases for a gateway:

- The gateway is trusted with the user’s keys. In this case, the gateway can decrypt the message, possibly add a note that it has done so, and gateway the unencrypted form, possibly applying X.400 security functions, and possibly attaching a copy of the original, encrypted material for reference. This does nothing to protect the transfer from gateway to recipient, unless suitable X.400-native security is applied. It also means that the gateway must be part of the user’s trusted environment.

- The gateway is not trusted with the recipient’s keys. In this case, encapsulation is the only approach that preserves any information at all.

The valid options for a MIXER gateway are therefore:

- Decrypt the body part

- Encapsulate the body part

- Drop the body part

The MIXER WG has shown strong preference for the encapsulation alternative, and urges anyone who thinks of buying or implementing gateway decryption to carefully evaluate this choice in light of the company's general security policy.

## 8. Conformance requirements

In order to be called MIXER conformant, a gateway must implement:

- Encapsulation of MIME content in the FTBP body part
- Encapsulation of X.400 body parts in the x400-bp body part
- Encapsulation of FTBP body parts in the application/x-ftbp.oid body part
- Encapsulation of security multiparts using HARPOON
- Text/plain <-> IA5Text
- Text/plain; charset=iso-8859-* <-> GeneralText
- Multipart/* <-> ForwardedIPMessage
- message/RFC822 <-> ForwardedIPMessage
- application/octet-stream <-> FTBP unknown
- application/octet-stream <-> BilaterallyDefined
- A configuration choice of which application/octet-stream translation to use

All other parts of this specification MAY be implemented by the gateway. If they are implemented at all, they MUST be implemented conformant to this specification.

In this context, a feature is "implemented" in a product if it is possible to configure the product in such a way that this feature is used. This specification does not restrict the product to only be configured in such a fashion.

## 9. Security Considerations

The security issues identified in this memo are:

(1) Security implications of using filenames that arrive in body part headers (section 2.3.2)

(2) Security implications of letting a gateway handle encrypted and/or signed content (section 7.3 and 7.4)

If a gateway fetches message/external-body on behalf of the recipient, as described in section 7.1, it may be tricked into performing inappropriate actions by malicious senders.

In addition, all the normal caveats that apply to sending data that may contain executable code apply to UAs on both sides of the gateway.

## 10. Author's Address

Harald Tveit Alvestrand
UNINETT
P.O.box 6883 Elgeseter
N-7002 Trondheim
NORWAY

EMail: Harald.T.Alvestrand@uninett.no

## 11. Acknowledgements

The author wishes to thank all the members of the MIXER WG for their valuable input, and in particular (in no particular order):

Steve Kille, Peter Sylvester, Ned Freed, Julian Onions, Ruth Moulton, Keith Moore, Alain Zahm, Urs Eppenberger, Kevin Jordan, Jeroen Houttuin, Claudio Allocchio, Colin Robbins, Steven Thomson, Jim Craigie, Efifiom Edem, David Wilson, and many others who have been active over the long lifetime of this document.

## References

[RFC-822]
Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, August, 1982.

[MIME]
Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[MIME-HDR]
Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

[HARPOON]
Alvestrand, H., Romaguera, J., and K. Jordan, "Rules for downgrading messages from X.400/88 to X.400/84 when MIME content-types are present in the messages", RFC 1496, August 1993.

[MIMETRANS]
Vaudreuil, G., "Transition of Internet Mail from Just-Send-8 to 8Bit-SMTP/MIME", RFC 1428, February 1993.

[MIXER]
Kille, S., "Mapping between X.400(1988) / ISO 10021 and RFC-822", RFC 1327, May 1992.

[T.4]
CCITT Recommendation T.4, Standardization of Group 3 Facsimile Apparatus for Document Transmission (1988)

[T.30]
CCITT Recommendation T.30, Procedures For Document Facsimile Transmission in the General Switched Telephone Network (1988)

[T.411]
CCITT Recommendation T.411 (1988), Open Document Architecture (ODA) and Interchange Format, Introduction and General Principles

[MOTIS]
ISO/IEC International Standard 10021, Information technology - Text Communication - Message-Oriented Text Interchange Systems (MOTIS) (Parts 1 to 8)

[X.400]
CCITT, Data Communication Networks - Message Handling Systems - Recommendations X.400 - X.420 (1988 version)

[X.420]
CCITT Recommendation X.420 (1988), Interpersonal Messaging System

[RFC-X400USE]
Alvestrand, H., "X.400 use of extended Character Sets", RFC 1502, August 1993.

[MAWG]
Electronic Messaging Association Message Attachment Working Group (MAWG): File Transfer Body Part Feasibility Project Guide - version 1.5 - September 1995

[CDISP]
Troost, R., and S. Dorner, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header", RFC 1806, June 1995.

[POSTSCRIPT]
Alvestrand, H., "Carrying PostScript in X.400 and MIME", RFC 2160, June 1997.

[IMAGES]
Alvestrand, H., "X.400 Image Body Parts", RFC 2158, June 1997.

[ODA]
Alvestrand, H., "A MIME Body Part for ODA", RFC 2161, June 1997.

[ISO 2022]
ISO/IEC 2022:1994(E): Information technology - Character code structure and extension techniques

[ISO 8859]
ISO 8859: Information processing -- 8-bit single-byte coded graphic character sets (various parts)

[2022-JP]
Murai, J., Crispin, M., and E. van der Poel, "Japanese Character Encoding for Internet Messages", RFC 1468, June 1993.

[MUST]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

APPENDIXES

Appendix A: Escape code normalization

The algorithm given here in pseudocode will reduce a GeneralString ISO-2022 unlimited use of shifts sequence to a pure 8-bit sequence that does not use shift sequences, if possible.

Some error conditions, like EOF, are not tested for. It crashes if asked to do something it cannot. Control character set switching is missing.

A similar routine, albeit more complex, can be written for normalizing to the ISO-2022-JP character set.

```
BEGIN: (from X.209)
  g0 = 6 (should be 2, but ignore the difference)
  g1 = NULL
  g2 = NULL
  g3 = NULL
  c0 = 1 (ASCII control)
  c1 = NULL
  leftset = &g0 (current input set, low)
  rightset = &g1 (current input set, high)
  lowset = 6 (output set, low)
  highset = NULL (output set, high)
  charset = US-ASCII

  (Init for the set tables)
  chartoid[{2D,2E,2F}, 41] = 100
  .....
  idtoname[100] = "ISO-8859-1"
  .....

WHILE (more data)
  CASE head of input
    {These are the locking shift sequences}
    INCASE "00/14": (LS0, SO)
        leftset = &g0;
    INCASE "00/15": (LS1, SI)
        leftset = &g
    INCASE "ESC 07/14": (LS1R)
        rightset = &g1;
    INCASE "ESC 07/13": (LS2R)
        rightset = &g2;
    INCASE "ESC 07/12": (LS3R)
        rightset = &g3;
    {There is missing code for handling the single shift function}
```

```
      {These are the changes of graphic character sets}
      {Note that G0 can contain only 94-character charsets}
      INCASE "ESC 28"
          g0 = chartoid[lastchar, next character]
          sethiset(g0)
      INCASE "ESC 2D", "ESC 29"
          g1 = chartoid[lastchar, next character]
          sethiset(g1)
      INCASE "ESC 2E", "ESC 2A"
          g2 = chartoid[lastchar, next character]
          sethiset(g2)
      INCASE "ESC 2F", "ESC 2B"
          g3 = chartoid[lastchar, next character]
          sethiset(g3)
      {control characters. There is missing code for changing these}
      INCASE 00/00-01/15 {normal control}
          write(char)
      INCASE 08/00-09/15 {upper control}
          write(char)
      {Normal characters}
      INCASE 02/00-07/15 (Left)
          IF (*leftset == lowset)
              write(char)
          ELSIF (*leftset == highset)
              write(char+80)
          ELSE
              ERROR "Shift error"
          ENDIF
      INCASE 10/00-15/15
          IF (*rightset == highset)
              write(char)
          ELSIF (*rightset == lowset)
              write(char-80)
          ELSE
              ERROR "Shift error"
          ENDIF
    ENDCASE
  ENDWHILE

   SUBROUTINE sethighset(g1)

          IF (highset == NULL)
              charset = idtoname[g1]
              highset = g1
          ELSIF (highset == g1)
              (it's OK)
          ELSE
              ERROR "Too many charsets encountered"
```

```
            ENDIF

   ENDROUTINE
```

## Appendix B: OID Assignments

```
   MIXER-MAPPINGS DEFINITIONS ::= BEGIN
   EXPORTS -- everything --;

   IMPORTS

      mixer -- { iso(1) org(3) dod(6) internet(1) mail(7) mixer(1) }
           FROM MIXER --Companion RFC--;

   mixer-headings OBJECT IDENTIFIER ::=
           { mixer 1 } -- called mime-mhs-headings in RFC 1495 --

   mixer-bodies OBJECT IDENTIFIER ::=
           { mixer 2 } -- called mime-mhs-bodies in RFC 1495 --

   -- mixer-core is defined as { mixer core(3) } in [MIXER]

   mixer-bp-data OBJECT IDENTIFIER ::=
           { mixer-bodies 1 }; -- called mime-mhs-bp-data in RFC 1494 --

   mixer-bp-parameter OBJECT IDENTIFIER ::=
           { mixer-bodies 2 };

   id-mime-bp-data OBJECT IDENTIFIER ::=
           { mixer-bp-data 1 };
   -- for debugging: mixer-bp-data is 1.3.6.1.7.1.2.1.1

   id-mime-bp-parameters OBJECT IDENTIFIER ::=
           { mixer-bp-parameter 1 };

   -- the following assignments were done in RFC 1494, using
   -- slightly different names, but the same numbers.
   -- their defining text is now is now in other documents
      id-mime-postscript-body OBJECT IDENTIFIER ::=
                    { mixer-bp-data 2 }

      id-mime-jpeg-body OBJECT IDENTIFIER ::=
                    { mixer-bp-data 3 }

      id-mime-gif-body OBJECT IDENTIFIER ::=
                    { mixer-bp-data 4 }

   -- This is a new definition, and defines an FTAM application
   reference,
   -- not a BP15 data OID.
   id-mime-ftbp-data OBJECT IDENTIFIER ::=
                     { mixer-bp-data 5 }
```

```
   -- The following heading extensions are defined
   id-hex-partial-message OBJECT IDENTIFIER ::=
              { mixer-headings 1 }

   id-hex-multipart-message OBJECT IDENTIFIER ::=
              { mixer-headings 2 } -- from RFC 1495; obsolete

   id-hex-multipart-message-v2 OBJECT IDENTIFIER ::=
            { mixer-headings 3 }

   END
```

## Appendix C: Registration information for the Teletex character set

The Teletex character set is a character set in which the ISO 2022 character set switching mechanism may be used to switch between the following registered ISO character sets:

ISO-IR-87 - JIS_C6226-1983; a 16-bit Japanese character set
ISO-IR-102 - a fairly standard US-ASCII variant
ISO-IR-103 - Latin characters using non-spacing accents
ISO-IR-106 - Control characters for C0 use; CR, LF, FF and a few more.
ISO-IR-107 - Control characters for C1 use

Its intended use of this character set is to represent data that comes from ISO protocols that use the ASN.1 construct "TeletexString" or "T61string" without conversion.

The set of allowed character sets can be found in CCITT recommendation X.208(1988), chapter 31.2 and Table 6/X.208.

The rules for encoding the data type can be found in CCITT recommendation X.209(1988), chapter 23. It states that at the beginning of the string, G0 is always ISO-IR-102, C0 is ISO-IR-106, and C1 is ISO-IR-107.

The specification seems somehow to have missed the implicit assumption that ISO-IR-103 is designated and invoked as G1 and shifted into the upper half of the character set which seems to be assumed at least by the X.400 and X.500 software that uses TeletexStrings; implementors should act as if the sequence ESC 2/9 7/6 LS1R is always present at the beginning of the data; however, when generating Teletex strings, implementors should include the sequence ESC 2/9 7/6 within the string before the first occurence of a character from ISO-IR-103.

The rules for interpreting T.61 data are found (I believe) in CCITT recommendations T.51, T.52 and T.53 (data from the ITU WWW server):

T.51 (09/92) [Rev.1] [26 pp.] [Publ.: May.93]
Latin based coded character sets for telematic services
T.52 (1993) [New] [88 pp.] [Publ.: Apr.94]
Non-Latin coded character sets for telematic services
T.53 (04/94) [New] [68 pp.] [Publ.: Jan.95]
Character coded control functions for telematic services

The Teletex character set is closely related to (but not identical with) that specified in ISO 6937.

No further restrictions are imposed by this registration; in particular, character set switching can occur anywhere, and there is no guarantee that the character sets will be switched "back" at the end.

Appendix D: IANA Registration form for new mappings

To: IANA@isi.edu
Subject: Registration of new X.400/MIME content type mapping

MIME type name:

(this must have been registered previously with IANA)

X.400 body part:

IF BP15:

- X.400 Object Identifier for Data:

(If left empty, an OID will be assigned by IANA under mixer-bp-data)

- X.400 Object Identifier for Parameters:

(If left empty, an OID will be assigned by IANA under mixer-bp-parameter. If it is not used, fill in the words NOT USED.)

X.400 ASN.1 Syntax:

(must be an EXTENDED-BODY-PART-TYPE macro, or reference to a Basic body part type)

IF FTBP:

- FTAM Object Identifier for application-reference:

- FTAM Object Identifier for contents-type:

(if left empty, unstructured-binary is assumed)

Conversion algorithm:

(must be defined completely enough for independent implementation. It may be defined by reference to RFCs).
Person & email address to contact for further information:

INFORMATION TO THE SUBMITTER:

The accepted registrations will be listed in the "Assigned Numbers" series of RFCs. The information in the registration form is freely distributable.

Full Copyright Statement