UCL Technical Report 120
Mailgroup Note 19

Network Working Group S.E. Kille
Request for Comments: 987 University College London
June 1986

Mapping between X.400 and RFC 822

Status of This Memo

This RFC suggests a proposed protocol for the ARPA-Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

This document describes a set of mappings which will enable interworking between systems operating the CCITT X.400 (1984) series of protocols [CCITT84a], and systems using the RFC 822 mail protocol [Crocker82a], or protocols derived from RFC 822. The approach aims to maximise the services offered across the boundary, whilst not requiring unduly complex mappings. The mappings should not require any changes to end systems.

This specification should be used when this mapping is performed on the ARPA-Internet or in the UK Academic Community. This specification may be modified in the light of implementation experience, but no substantial changes are expected.

Chapter 1 -- Overview

1.1. X.400

The X.400 series protocols have been defined by CCITT to provide an Interpersonal Messaging Service (IPMS), making use of a store and forward Message Transfer Service. It is expected that this standard will be implemented very widely. As well as the base standard (X.400), work is underway on various functional standards of profiles which specify how X.400 will be used in various communities. Many of the major functional standards (e.g. from CEPT, CEN/CENELEC, and NBS) are likely to be similar. Some of the decisions in this document are in the light of this work. No reference is given, as these documents are not currently stable.

1.2. RFC 822

RFC 822 evolved as a messaging standard on the DARPA (the US Defense Advanced Research Projects Agency) Internet. It is currently used on the ARPA-Internet in conjunction with two other standards: RFC 821, also known as Simple Mail Transfer Protocol (SMTP) [Postel82a], and RFC 920 which is a specification for a domain name system and a distributed name service [Postel84a]. RFC 822, or protocols derived from RFC 822 are used in a number of other networks. In particular:

UUCP Networks

UUCP is the UNIX to UNIX CoPy protocol <0>, which is usually used over dialup telephone networks to provide a simple message transfer mechanism. There are some extensions to RFC 822, particularly in the addressing. They are likely to use domains which conform to RFC 920, but not the corresponding domain nameservers [Horton86a].

CSNET

Some portions of CSNET will follow the ARPA-Internet protocols. The dialup portion of CSNET uses the Phonenet protocols as a replacement for RFC 821. This portion is likely to use domains which conform to RFC 920, but not the corresponding domain nameservers.

BITNET

Some parts of BITNET use RFC 822 related protocols, with EBCDIC encoding.

JNT Mail Networks

A number of X.25 networks, particularly those associated with the UK Academic Community, use the JNT (Joint Network Team) Mail Protocol, also known as Greybook [Kille84a]. This is used with domains and name service specified by the JNT NRS (Name Registration Scheme) [Larmouth83a].

The mappings specified here are appropriate for all of these networks.

1.3. The Need for Conversion

There is a large community using RFC 822 based protocols for mail services, who will wish to communicate with X.400 systems. This will be a requirement, even in cases where communities intend to make a transition to use of X.400, where conversion will be needed to ensure a smooth service transition. It is expected that there will be more than one gateway <1>, and this specification will enable them to behave in a consistent manner. These gateways are sometimes called mail relays. Consistency between gateways is desirable to provide:

1. Consistent service to users.

2. The best service in cases where a message passes through multiple gateways.

1.4. General Approach

There are a number of basic principles underlying the details of the specification.

1. The specification should be pragmatic. There should not be a requirement for complex mappings for 'Academic' reasons. Complex mappings should not be required to support trivial additional functionality.

2. Subject to 1), functionality across a gateway should be as high as possible.

3. It is always a bad idea to lose information as a result of any transformation. Hence, it is a bad idea for a gateway to discard information in the objects it processes. This includes requested services which cannot be fully mapped.

4. All mail gateways actually operate at exactly one level

above the layer on which they conceptually operate. This implies that the gateway must not only be cognisant of the semantics of objects at the gateway level, but also be cognisant of higher level semantics. If meaningful transformation of the objects that the gateway operates on is to occur, then the gateway needs to understand more than the objects themselves.

1.5. Gatewaying Model

1.5.1. X.400

The CCITT X.400 series recommendations specify a number of services and protocols. The services are specified in X.400. Two of these services are fundamental to this document:

1. The Message Transfer Service, which can be provided by either the P1 or P3 protocols, which are specified in X.411 [CCITT84b]. This document talks in terms of P1, but the mappings are equally applicable to P3.

2. The Interpersonal Messaging Service (IPMS), which is provided by the P2 protocol specified in X.420 [CCITT84c].

This document considers only IPMS, and not of any other usage of the Message Transfer Service. This is reasonable, as RFC 822, broadly speaking, provides a service corresponding to IPMS, and no services other than IPMS have been defined over the Message Transfer Service. As none of the RTS (Reliable Transfer Service) service elements is available to the IPMS user, this level and lower levels are of no concern in this gatewaying specification. Note that in this memo "IP" means "InterPersonal" (not Internet Protocol).

The Message Transfer Service defines an end-to-end service over a series of Message Transfer Agents (MTA). It also defines a protocol, P1, which is used between a pair of MTAs. This protocol is simply a file format (Message Protocol Data Unit, or MPDU), transferred between two MTAs using the RTS. There are three types of MPDU:

User MPDU

This contains envelope information, and uninterpreted contents. The envelope includes an ID, an originator, a

list of recipients, and trace information. It is used to carry data for higher level services.

Probe

This contains only envelope information. It is used to determine whether a User UMPDU could be delivered to a given O/R (originator/recipient) name.

Delivery Report

This contains envelope information, and specified contents. It is used to indicate delivery success or failure of a User or Probe MPDU over the Message Transfer Service.

IPMS (P2) specifies two content types for the P1 User MPDU (User Agent Protocol Data Units or UAPDU):

Interpersonal Message (IM-UAPDU)

This has two components: a heading, and a body. The body is structured as a sequence of body parts, which may be basic components (e.g.IA5 text, or G3 fax), or IP Messages. The header contains end to end user information, such as subject, primary recipients (To:), and priority. The validity of these fields is not guaranteed by the Message Transfer Service. This provides the basic IPMS.

Status Report (SR-UAPDU)

This UAPDU has defined contents. It is used to indicate that a message has been received by a User Agent. It does not have to be implemented.

1.5.2. RFC 822

RFC 822 is based on the assumption that there is an underlying service, which is here called the 822-P1 service. The 822-P1 service provides three basic functions:

1. Identification of a list of recipients.

2. Identification of an error return address.

3. Transfer of an RFC 822 message.

It is possible to achieve 2) within the RFC 822 header. Some 822-P1 protocols, in particular SMTP, can provide additional functionality, but as these are neither mandatory in SMTP, nor available in other 822-P1 protocols, they are not considered here. Details of aspects specific to a number of 822-P1 protocols are given in appendices B to E. An RFC 822 message consists of a header, and content which is uninterpreted ASCII text. The header is divided into fields, which are the protocol elements. Most of these fields are analogous to P2 header elements, although some are analogous to P1 envelope elements.

1.5.3. The Gateway

Given this functional description of the two protocols, the functional nature of a gateway can now be considered. It would be elegant to consider the 822-P1 service mapping onto P1 and RFC 822 mapping onto P2, but reality just does not fit. Therefore one must consider that P1 or P1 + P2 on one side are mapped into RFC 822 + 822-P1 on the other in a slightly tangled manner. The details of the tangle will be made clear in chapter 5. The following basic mappings are thus proposed. When going from RFC 822 to X.400, an RFC 822 message and the associated 822-P1 information is always mapped into an IM-UAPDU and the associated P1 envelope. Going from X.400 to RFC 822, an RFC 822 message and the associated 822-P1 information may be derived from:

1. A Delivery Report MPDU

2. An SR-UAPDU and the associated P1 envelope.

3. An IM-UAPDU and the associated P1 envelope.

Probe MPDUs must be processed by the gateway - this is discussed in chapter 5. Any other User MPDUs are not mapped by the gateway, and should be rejected at the gateway.

1.6. Document Structure

This document has five chapters:

1. Overview - this document.

2. Service Elements - This describes the (end user) services mapped by a gateway.

3. Basic mappings - This describes some basic notation used in chapters 3-5, the mappings between character sets, and some fundamental protocol elements.

4. Addressing - This considers the mapping between X.400 O/R names and RFC 822 addresses, which is a fundamental gateway component.

5. Protocol Elements - This describes the details of all other mappings.

There are also six appendices:

A. Quoted String Encodings.

B. Mappings Specific to JNT Mail.

C. Mappings Specific to Internet Mail.

D. Mappings Specific to Phonenet Mail.

E. Mappings Specific to UUCP Mail.

F. Format of Address Tables.

1.7. Acknowledgements

This document is eclectic, and credit should be given:

- Study of the EAN X.400 system code which performs this function [Neufeld85a]. Some detailed clarification was made by the DFN report on EAN [Bonacker85a].

- An unpublished ICL report, which considered a subset of the problem [ICL84a].

- A document by Marshall Rose [Rose85a].

- A document by Mark Horton [Horton85a]. The string encodings of chapter 3 were derived directly from this work, as is much of chapter 4.

- Discussion on a number of electronic mailing lists.

- Meetings in the UK and the US.

Chapter 2 -- Service Elements

RFC 822 and X.400 provide a number of services to the end user. This document describes the extent to which each service can be supported across an X.400 <-> RFC 822 gateway. The cases considered are single transfers across such a gateway, although the problems of multiple crossings are noted where appropriate.

When a service element is described as supported, this means that when this service element is specified by a message originator for a recipient behind a gateway, that it is mapped by the gateway to provide the service implied by the element. For example, if an RFC 822 originator specifies a Subject: field, this is considered to be supported, as an X.400 recipient will get a subject indication. Support implies:

- Semantic correspondence.

- No loss of information.

- Any actions required by the service element.

For some services, the corresponding protocol elements map well, and so the service can be fully provided. In other cases, the service cannot be provided, as there is a complete mismatch. In the remaining cases, the service can be partially fulfilled. The level of partial support is summarised.

NOTE: It should be clear that support of service elements on reception is not a gatewaying issue. It is assumed that all outbound messages are fully conforming to the appropriate standards.

2.1. RFC 822

RFC 822 does not explicitly define service elements, as distinct from protocol elements. However, all of the RFC 822 header fields, with the exception of trace, can be regarded as corresponding to implicit RFC 822 service elements. A mechanism of mapping used in several cases, is to place the text of the header into the body of the IP Message. This can usually be regarded as partial support, as it allows the information to be conveyed to the end user even though there is no corresponding X.400 protocol element. Support for the various service elements (headers) is now listed.

Date:

Supported.

From:

Supported. For messages where there is also a sender field, the mapping is to "Authorising Addresses", which has subtly different semantics to the general RFC 822 usage of From:.

Sender:

Supported.

Reply-To:

Supported.

To:

Supported.

Cc:

Supported.

Bcc:

Supported.

Message-Id:

Supported.

In-Reply-To:

Supported, for a single reference in msg-id form. Other cases are passed in the message text.

References:

Supported.

Keywords:

Passed in the message text.

Subject:

Supported.

Comments:

Passed in the message text.

Encrypted:

Passed in the message text. This may not be very useful.

Resent-*

Passed in the message text. In principle, these could be supported in a fuller manner, but this is not suggested.

Other Fields

In particular X-* fields, and "illegal" fields in common usage (e.g. "Fruit-of-the-day:") are passed in the message text.

2.2. X.400

When mapping from X.400 to RFC 822, it is not proposed to map any elements into the body of an RFC 822 message. Rather, new RFC 822 headers are defined. It is intended that these fields will be registered, and that co-operating RFC 822 systems may use them. Where these new fields are used, and no system action is implied, the service can be regarded as being almost supported. Chapter 5 describes how to map these new headers in both directions. Other elements are provided, in part, by the gateway as they cannot be provided by RFC 822. Some service elements are are marked N/A (not applicable). These elements are only applicable to User Agent / Message Transfer Agent interaction and have no end-to-end implication. These elements do not need to be mapped by the gateway.

2.2.1. Message Transfer Service Elements

Access Management

N/A.

Content Type Indication

Not mapped. As it can only have one value (P2), there is little use in creating a new RFC 822 header field, unless it was to distinguish delivery reports.

Converted Indication

Supported by a new RFC 822 header.

Delivery Time Stamp Indication

N/A.

Message Identification

Supported, by use of a new RFC 822 header. This new header is required, as X.400 has two message-ids whereas RFC 822 has only one.

Non-delivery Notification

Not supported, although in general an RFC 822 system will return errors as IP messages. In other elements, this pragmatic result is treated as effective support of this service element.

Original Encoded Information Types Indication

Supported as a new RFC 822 header.

Registered Encoded Information Types

N/A.

Submission Time Stamp Indication

Supported.

Alternate Recipient Allowed

Not supported. Any value is ignored by the gateway.

Deferred Delivery

Support is optional. The framework is provided so that messages may be held at the gateway. However, a gateway

following this specification does not have to do this. This is in line with the emerging functional standards.

Deferred Delivery Cancellation

Supported.

Delivery Notification

Supported at gateway. Thus, a notification is sent by the gateway to the originator <2>.

Disclosure of Other Recipients

Supported by use of a new RFC 822 header.

Grade of Delivery Selection

Supported as a new RFC 822 header. In general, this will only be for user information in the RFC 822 world.

Multi-Destination Delivery

Supported.

Prevention of Non-delivery Notification

Not Supported, as there is no control in the RFC 822 world (but see Non-delivery Notification).

Return of Contents

This is normally the case, although the user has no control (but see Non-delivery Notification).

Conversion Prohibition

Supported. Note that in practice this support is restricted by the nature of the gateway.

Explicit Conversion

Supported, for appropriate values (See the IPMS Typed Body service element).

Implicit Conversion

Supported, in the sense that there will be implicit conversion to IA5 in cases where this is practical.

Probe

Supported at the gateway (i.e. the gateway services the probe).

Alternate Recipient Assignment

N/A.

Hold for Delivery

N/A.

2.2.2. Interpersonal Message Service Elements

IP-message Identification

Supported.

Typed Body

Supported. IA5 is fully supported. ForwardedIPMessage is supported, with some loss of information. A subset of TTX is supported (see section 5 for the specification of this subset), with some loss of information. SFD may be supported, with some loss of information. TTX and SFD are only supported when conversion is allowed. Other types are not supported.

Blind Copy Recipient Indication

Supported.

Non-receipt Notification

Not supported.

Receipt Notification

Not supported.

Auto-forwarded Indication

Supported as new RFC 822 header.

Originator Indication

Supported.

Authorising User's Indication

Supported, although the mapping (From:) is not quite the same.

Primary and Copy Recipients Indication

Supported.

Expiry Date Indication

Supported as new RFC 822 header. In general, only human action can be expected.

Cross Referencing Indication

Supported.

Importance Indication

Supported as new RFC 822 header.

Obsoleting Indication

Supported as new RFC 822 header.

Sensitivity Indication

Supported as new RFC 822 header.

Subject Indication

Supported.

Reply Request Indication

Supported as comment next to address.

Forwarded IP-message Indication

Supported, with some loss of information.

Body Part Encryption Indication

Not supported.

Multi-part Body

Supported, with some loss of information, in that the structuring cannot be formalised in RFC 822.

Chapter 3 -- Basic Mappings

3.1. Notation

The P1 and P2 protocols are encoded in a structured manner according to the X.409 specifications, whereas RFC 822 is text encoded. To define a detailed mapping, it is necessary to refer to detailed protocol elements in each format. This is described.

3.1.4. RFC 822

Structured text is defined according to the Extended Backus Naur Form (EBNF) defined in section 2 of RFC 822 [Crocker82a]. In the EBNF definitions used in this specification, the syntax rules given in Appendix D of RFC 822 are assumed. When these EBNF tokens are referred to outside an EBNF definition, they are identified by the string "882." appended to the beginning of the string (e.g. 822.addr-spec). Additional syntax rules, to be used throughout this specification are defined in this chapter.

The EBNF is used in two ways.

1. To describe components of RFC 822 messages (or of 822-P1 components). In this case, the lexical analysis defined in section 3 of RFC 822 should be used. When these new EBNF tokens are referred to outside an EBNF definition, they are identified by the string "EBNF." appended to the beginning of the string (e.g. EBNF.bilateral-info).

2. To describe the structure of IA5 or ASCII information not in an RFC 822 message. In these cases, tokens will either be self delimiting, or be delimited by self delimiting tokens. Comments and LWSP are not used as delimiters.

3.1.5. X.409

An element is referred to with the following syntax, defined in EBNF:

```
element        = protocol "." definition *( "." definition )
protocol       = "P1" / "P2"
definition     = identifier / context
identifier     = ALPHA *< ALPHA or DIGIT or "-" >
context        = "[" 1*DIGIT "]"
```

For example, P2.Heading.subject defines the subject element of the P2 heading. The same syntax is also used to refer to element values. For example, P1.EncodedInformationTypes.[0].g3Fax refers to a value of P1.EncodedInformationTypes.[0] .

3.2. ASCII and IA5

A gateway will interpret all IA5 as ASCII. Thus, they are treated identically for the rest of this document.

3.3. Universal Primitives

There is a need to convert between ASCII text, and some of the Universal Primitive types defined in X.409 [CCITT84d]. For each case, an EBNF syntax definition is given, for use in all of this specification. All EBNF syntax definitions of Universal Primitives are in lower case, whereas X.409 primitives are referred to with the first letter in upper case. Except as noted, all mappings are symmetrical.

3.3.1. Boolean

Boolean is encoded as:

```
boolean = "TRUE" / "FALSE"
```

3.3.2. NumericString

NumericString is encoded as:

```
numericstring = *DIGIT
```

3.3.3. PrintableString

PrintableString is a restricted IA5String defined as:

```
printablestring  = *( ps-char / ps-delim )

ps-char          = 1DIGIT /  1ALPHA / " " / "'" / "+" / ")"
                   / "," / "-" / "." / "/" / ":" / "=" / "?"

ps-delim         = "("
```

A structured subset of EBNF.printablestring is now defined. This can be used to encode ASCII in the PrintableString character set.

```
ps-encoded       = *( ps-char / ps-encoded-char )

ps-encoded-char  =   "(a)"               ; (@)
                   / "(p)"               ; (%)
                   / "(b)"               ; (!)
                   / "(q)"               ; (")
                   / "(u)"               ; (_)
                   / "(" 3DIGIT ")"
```

The 822.3DIGIT in EBNF.ps-encoded-char must have range 0-127 (Decimal), and is interpreted in decimal as the corresponding ASCII character. Special encodings are given for: at sign (@), percent (%), exclamation mark/bang (!), double quote ("), and underscore (_). These characters are not included in PrintableString, but are common in RFC 822 addresses. The abbreviations will ease specification of RFC 822 addresses from an X.400 system.

An asymmetric mapping between PrintableString and ASCII can now be defined <3>. To encode ASCII as PrintableString, the EBNF.ps-encoded syntax is used, with all EBNF.ps-char AND EBNF.ps-delim mapped directly <4>. All other 822.CHAR are encoded as EBNF.ps-encoded-char. There are two cases of encoding PrintableString as ASCII. If the PrintableString can be parsed as EBNF.ps-encoded, then the previous mapping should be reversed. If not, it should be interpreted as EBNF.printablestring.

Some examples are now given. Note the arrows which indicate asymmetrical mappings:

```
PrintableString           ASCII

'a demo.'         <->   'a demo.'
foo(a)bar         <->   foo@bar

(q)(u)(p)(q)      <->   "_%"
(a)               <->   @
(a)               <-    (a)
(040)a(041)        ->    (a)
(040)(a)           ->    (@
((a)              <-    (@
```

The algorithm is designed so that it is simple to use in all common cases, so that it is general, and so that it is straightforward to code. It is not attempting to minimise the number of pathological cases.

3.3.4. T.61String

T.61 strings are, in general, only used for conveying human interpreted information. Thus, the aim of a mapping should be to render the characters appropriately in the remote character set, rather than to maximise reversibility. The mappings defined in the CEN/CENELEC X.400 functional standard should be used [CEN/CENELEC/85a]. These are based on the mappings of X.408 (sections 4.2.2 and 5.2.2).

3.3.5. UTCTime

Both UTCTime and the RFC 822 822.date-time syntax contain: Year (lowest two digits), Month, Day of Month, hour, minute, second (optional), and Timezone. 822.date-time also contains an optional day of the week, but this is redundant. Therefore a symmetrical mapping can be made between these constructs <5>. The UTCTime format which specifies the timezone offset should be used, in line with CEN/CENELEC recommendations.

Chapter 4 -- Addressing

Addressing is probably the trickiest problem of an X.400 <-> RFC 822 gateway. Therefore it is given a separate chapter. This chapter, as a side effect, also defines a standard textual representation of X.400 addresses.

Initially we consider an address in the (human) mail user sense of "what is typed at the mailsystem to reference a human". A basic RFC 822 address is defined by the EBNF EBNF.822-address:

```
822-address     = [ route ] addr-spec
```

In an 822-P1 protocol, the originator and each recipient should be considered to be defined by such a construct. In an RFC 822 header, the EBNF.822-address is encapsulated in the 822.address syntax rule, and there may also be associated comments. None of this extra information has any semantics, other than to the end user.

The basic X.400 address is defined by P1.ORName. In P1 all recipient P1.ORnames are encapsulated within P1.RecipientInfo, and in P2 all P2.ORNames <6> are encapsulated within P2.ORDescriptor.

It can be seen that RFC 822 822.address must be mapped with P2.ORDescriptor, and that RFC 822 EBNF.822-address must be mapped with P1.ORName (originator) and P1.RecipientInfo (recipients).

This chapter is structured as follows:

4.1 Introduction.

4.2 A textual representation of P1.ORName. This is needed for the later mappings, and as a side effect provides a standard representation for O/R names.

4.3 Mapping between EBNF.822-address and P1.ORName

4.4 The Full P1 / 822-P1 Mapping

4.5 The Full P2 / RFC 822 Mapping

4.6 Mapping Message-IDs.

4.1. A textual representation of P1.ORName.

P1.ORName is structured as a set of attribute value pairs. It is clearly necessary to be able to encode this in ASCII for gatewaying purposes. A general encoding is given here, which may be used as a basis for a user interface, as well as for the defined gateway mapping.

4.1.1. Basic Representation

A series of BNF definitions of each possible attribute value pair is given, which is given a 1:1 mapping with the X.400 encoding. The rest of the mapping then talks in terms of these BNF components, with the mapping to X.400 encoding being trivial.

```
attributevalue = c / admd / prmd / x121 / t-id / o / ou
                 / ua-id / pn.g / pn.i / pn.s / pn.gq / dd.value

c        = printablestring       ; P1.CountryName
admd     = printablestring       ; P1.AdministrationDomainName
prmd     = printablestring       ; P1.PrivateDomainName
x121     = numericstring         ; P1.X121Address
t-id     = numericstring         ; P1.TerminalID
o        = printablestring       ; P1.OrganisationName
ou       = printablestring       ; P1.OrganisationalUnit
ua-id    = numericstring         ; P1.UniqueUAIdentifier
pn.s     = printablestring       ; P1.PersonalName.surName
pn.g     = printablestring       ; P1.PersonalName.givenName
pn.i     = printablestring       ; P1.PersonalName.initials
pn.gq    = printablestring       ; P1.PersonalName.generation
                                   Qualifier
dd.value = printablestring       ; P1.DomainDefined
                                   Attribute.value
```

In cases where an attribute can be encoded as either a PrintableString or NumericString (Country, ADMD, PRMD) it is assumed that the NumericString encoding will be adopted if possible. This prevents the encoding of PrintableString where the characters are all numbers. This restriction seems preferable to the added complexity of a general solution. Similarly, we can define a set of attribute types.

```
dd.type = printablestring      ; P1.DomainDefinedAttribute.type

standard-type =
          "C"           ; P1.CountryName
        / "ADMD"        ; P1.AdministrationDomainName
        / "PRMD"        ; P1.PrivateDomainName
        / "X121"        ; P1.X121Address
        / "T-ID"        ; P1.TerminalID
        / "O"           ; P1.OrganisationName
        / "OU"          ; P1.OrganisationalUnit
        / "UA-ID"       ; P1.UniqueUAIdentifier
        / "S"           ; P1.PersonalName.surName
        / "G"           ; P1.PersonalName.givenName
        / "I"           ; P1.PersonalName.initials
        / "GQ"          ; P1.PersonalName.generationQualifier

standard-dd-type =
          "RFC-822"     ; dd.type = "RFC-822"
        / "JNT-Mail"    ; dd.type = "JNT-Mail"
        / "UUCP"        ; dd.type = "UUCP"
```

4.1.2. Encoding of Personal Name

Handling of Personal Name based purely on the EBNF.standard-type syntax defined above is likely to be clumsy. It seems desirable to utilise the "human" conventions for encoding these components. A syntax is proposed here. It is designed to cope with the common cases of O/R Name specification where:

1. There is no generational qualifier

2. Initials contain only letters <7>.

3. Given Name does not contain full stop ("."), and is at least two characters long.

4. If Surname contains full stop, then it may not be in the first two characters, and either initials or given name is present.

The following EBNF is defined:

```
encoded-pn      = [ given "." ] *( initial "." ) surname

given           = 2*<ps-char not including ".">

initial         = ALPHA

surname         = printablestring
```

Subject to the above restriction, this is a reversible mapping.

For example:

```
GivenName       = "Marshall"
Surname         = "Rose"

Maps with  "Marshall.Rose"

Initials        = "MT"
Surname         = "Rose"

Maps with  "M.T.Rose"

GivenName       = "Marshall"
Initials        = "MT"
Surname         = "Rose"

Maps with  "Marshall.M.T.Rose"
```

Note that CCITT guidelines suggest that Initials is used to encode ALL initials. Therefore, the proposed encoding is "natural" when either GivenName or Initials, but not both, are present. The case where both are present can be encoded, but this appears to be contrived!

4.1.3. Two encodings of P1.ORName

Given this structure, we can specify a BNF representation of an O/R Name.

```
   std-orname      = 1*( "/" attribute "=" value ) "/"
   attribute       = standard-type
                   / "PN"
                   / standard-dd-type
                   / registered-dd-type
                   / "DD." std-printablestring
   value           = std-printablestring
   registered-dd-type
                   = std-printablestring
   std-printablestring =
                   = *( std-char / std-pair )
   std-char        = <ps-delim, and any ps-char except "/"
                     and "=">
   std-pair        = "$" ( ps-delim / ps-char )
```

If the type is PN, the value is interpreted according to EBNF.encoded-pn, and the components of P1.PersonalName derived accordingly. If the value is registered-dd-type, if the value is registered at the SRI NIC as an accepted Domain Defined Attribute type, then the value should be interpreted accordingly. This restriction maximises the syntax checking which can be done at a gateway.

Another syntax is now defined. This is intended to be compatible with the syntax used for 822.domains. This syntax is not intended to be handled by users.

```
   dmn-orname      = dmn-part *( "." dmn-part )
   dmn-part        = attribute "$" value
   attribute       = standard-type
                   / "~" dmn-printablestring
   value           = dmn-printablestring
   dmn-printablestring =
                   = *( dmn-char / dmn-pair )
   dmn-char        = <ps-delim, and any ps-char except ".">
   dmn-pair        = "\."
```

For example: C$US.ADMD$ATT.~ROLE$Big\.Chief

4.2. Mapping between EBNF.822-address and P1.ORName

Ideally, the mapping specified would be entirely symmetrical and global, to enable addresses to be referred to transparently in the remote system, with the choice of gateway being left to the Message Transfer Service. There are two fundamental reasons why this is not possible:

1. The syntaxes are sufficiently different to make this awkward.

2. In the general case, there would not be the necessary administrative co-operation between the X.400 and RFC 822 worlds, which would be needed for this to work.

Therefore, an asymmetrical mapping is defined.

4.2.1. X.400 encoded in RFC 822

The std-orname syntax is used to encode O/R Name information in the 822.local-part of EBNF.822-address. Further O/R Name information may be associated with the 822.domain component. This cannot be used in the general case, basically due to character set problems, and lack of order in X.400 O/R Names. The only way to encode the full PrintableString character set in a domain is by use of the 822.domain-ref syntax. This is likely to cause problems on many systems. The effective character set of domains is in practice reduced from the RFC 822 set, by restrictions imposed by domain conventions and policy.

A generic 822.address consists of a 822.local-part and a sequence of 822.domains (e.g. <@domain1,@domain2:user@domain3>). All except the 822.domain associated with the 822.local-part (domain3 in this case) should be considered to specify routing within the RFC 822 world, and will not be interpreted by the gateway (although they may have identified the gateway from within the RFC 822 world). The 822.domain associated with the 822.local-part may also identify the gateway from within the RFC 822 world. This final 822.domain may be used to determine some number of O/R Name attributes. The following O/R Name attributes are considered as a hierarchy, and may be specified by the domain. They are (in order of hierarchy):

Country, ADMD, PRMD, Organisation, Organisational Unit

There may be multiple Organisational Units.

Associations may be defined between domain specifications, and some set of attributes. This association proceeds hierarchically: i.e. if a domain implies ADMD, it also implies country. If one of the hierarchical components is omitted from an X.400 structure, this information can be associated with the corresponding domain (e.g. a domain can be mapped onto a Country/ADMD/Organisation tuple). Subdomains under this are associated according to the O/R Name hierarchy. For example:

=> "AC.UK" might be associated with
C="234", ADMD="BT", PRMD="DES"

then domain "R-D.Salford.AC.UK" maps with
C="234", ADMD="BT", PRMD="DES", O="Salford", OU="R-D"

There are two basic reasons why a domain/attribute mapping might be maintained, as opposed to using simply subdomains:

1. As a shorthand to avoid redundant X.400 information. In particular, there will often be only one ADMD per country, and so it does not need to be given explicitly.

2. To deal with cases where attribute values do not fit the syntax:

```
domain-syntax   = ALPHA [ *alphanumhyphen alphanum ]
alphanum        = <ALPHA or DIGIT>
alphanumhyphen  = <ALPHA or DIGIT or HYPHEN>
```

Although RFC 822 allows for a more general syntax, this restriced syntax is chosen as it is the one chosen by the various domain service administrations.

This provides a general aliasing mechanism.

This set of mappings need only be known by the gateways relaying between the RFC 822 world, and the O/R Name namespace associated with the mapping in question. However, it is desirable (for the optimal mapping of third party addresses) for all gateways to know these mappings. A format for the exchange of this information is defined in Appendix F.

From the standpoint of the RFC 822 Message Transfer System, the domain specification is simply used to route the message in the

standard manner. The standard domain mechanisms are used to identify gateways, and are used to select appropriate gateways for the corresponding O/R Name namespace. In most cases, this will be done by registering the higher levels, and assuming that the gateway can handle the lower levels.

As a further mechanism to simplify the encoding of common cases, where the only attributes to be encoded on the LHS are Personal Name attributes which comply with the restrictions of 4.2.2, the 822.local-part may be encoded as EBNF.encoded-pn.

An example encoding is:

```
/PN=J.Linnimouth/GQ=5/@Marketing.Xerox.COM
```

encodes the P1.ORName consisting of

```
P1.CountryName                  = "US"
P1.AdministrationDomainName     = "ATT"
P1.OrganisationName             = "Xerox"
P1.OrganisationalUnit           = "Marketing"
P1.PersonalName.surName         = "Linnimouth"
P1.PersonalName.initials        = "J"
P1.PersonalName.GenerationQualifier = "5"
```

If the GenerationQualifier was not present, the encoding J.Linnimouth@Marketing.Xerox.COM could be used.

Note that in this example, the first three attributes are determined by the domain Xerox.COM. The OrganisationalUnit is determined systematically.

There has been an implicit assumption that an RFC 822 domain is either X.400 or RFC 822. This is pragmatic, but undesirable, as the namespace should be structured on a logical basis which does not necessarily correspond to the choice of Message Transfer protocols. The restriction can be lifted, provided that the nameservice deals with multiple message transfer protocols. This can happen in a straightforward manner for the UK NRS, as explained in [Kille86a]. It could also be achieved with the DARPA Domain Nameserver scheme by use of the WKS mechanism.

4.2.2. RFC 822 Encoded in X.400

In some cases, the encoding defined above may be reversed, to give a "natural" encoding of genuine RFC 822 addresses. This depends largely on the allocation of appropriate management domains.

The general case is mapped by use of domain defined attributes. Three are defined, according to the full environment used to interpret the RFC 822 information.

1. Domain defined type "RFC-822". This string is to be interpreted in the context of RFC 822, and RFC 920 [Crocker82a,Postel84a].

2. Domain defined type "JNT-Mail". This string is to be interpreted in the context of the JNT Mail protocol, and the NRS [Kille84a,Larmouth83a].

3. Domain defined type "UUCP". This is interpreted according to the constraints of the UUCP world [Horton86a].

These three are values currently known to be of use. Further recognised values may be defined. These will be maintained in a list at the SRI Network Information Center.

Other O/R Name attributes will be used to identify a context in which the O/R Name will be interpreted. This might be a Management Domain, or some part of a Management Domain which identifies a gateway MTA. For example:

1)

```
C               = "GB"
ADMD            = "BT"
PRMD            = "AC"
"JNT-Mail"      = "Jimmy(a)UK.CO.BT-RESEARCH-LABS"
```

2)

```
C               = "US"
ADMD            = "Telemail"
PRMD            = "San Fransisco"
O               = "U Cal"
OU              = "Berkeley"
"RFC-822"       = "postel(a)usc-isib.arpa"
```

Note in each case the PrintableString encoding of "@" as "(a)". In the first example, the "JNT-Mail" domain defined attribute is interpreted everywhere within the (Administrative or Private) Management Domain. In the second example, further attributes are needed within the Management Domain to identify a gateway. Thus, this scheme can be used with varying levels of Management Domain co-operation.

4.2.3. RFC 822 -> X.400

There are two basic cases:

1. X.400 addresses encoded in RFC 822. This will also include RFC 822 addresses which are given reversible encodings.

2. "Genuine" RFC 822 addresses.

The mapping should proceed as follows, by first assuming case 1).

STAGE 1.

1. If the 822-address is not of the form:

   local-part "@" domain

   go to stage 2.

2. Attempt to parse domain as:

   *(domain-syntax ".") known-domain

   Where known-domain is the longest possible match in a list of gatewayed domains. If this fails, and the domain does not explicitly identify the local gateway, go to stage 2. If it succeeds, allocate the attributes associated with EBNF.known-domain, and systematically allocate the attributes implied by each EBNF.domain-syntax component.

3. Map 822.local-part to ASCII, according to the definition of Appendix A. This step should be applied:

   A. If the source network cannot support 822.quoted-string (as discussed in Appendix A).

B. If the address is an 822-P1 recipient.

This mapping is always applied in case B, as it increases the functionality of the gateway, and does not imply any loss of generality. Mapping case B allows sites which cannot generate 822.quoted-string to address recipients the gateway, without the gateway having to know this explicitly. There is no loss of functionality, as the quoting character of Appendix A (#) is not in PrintableString. This seems desirable. It should not be applied in to other addresses, as a third party RFC#822 address containing the sequence EBNF.atom-encoded (as defined in Appendix A) would be transformed asymmetrically.

4. Map the result of 3) to EBNF.ps-encoded according to section 3.

5. Parse the result of 4) according to the EBNF EBNF.std-orname. If this parse fails, parse the result of 4) according to the EBNF EBNF.encoded-pn. If this also fails, go to stage 2. Otherwise, the result is a set of type/value pairs.

6. Associate the EBNF.attribute-value syntax (determined from the identified type) with each value, and check that it conforms. If not, go to stage 2.

7. Ensure that the set of attributes conforms both to the X.411 P1.ORName specification and to the restrictions on this set given in X.400. If not go to stage 2.

8. Build the O/R Name from this information.

STAGE 2.

This will only be reached if the RFC 822 EBNF.822-address is not a valid X.400 encoding. If the address is an 822-P1 recipient address, it must be rejected, as there is a need to interpret such an address in X.400. For the 822-P1 return address, and any addresses in the RFC 822 header, they should now be encoded as RFC 822 addresses in an X.400 O/R Name:

1. Convert the EBNF.822-address to PrintableString, as specified in chapter 3.

2. The domain defined attribute ("RFC-822", "JNT-Mail" or

"UUCP") appropriate to the gateway should be selected, and its value set.

3. Build the rest of the O/R Name in the local Management Domain agreed manner, so that the O/R Name will receive a correct global interpretation.

4.2.4. X.400 -> RFC 822

There are two basic cases:

1. RFC 822 addresses encoded in X.400.

2. "Genuine" X.400 addresses. This may include symmetrically encoded RFC 822 addresses.

When a P1 Recipient O/R Name is interpreted, gatewaying will be selected if there a single special domain defined attribute present ("RFC-822", "JNT-Mail" or "UUCP"). In this case, use mapping A. For other O/R Names which

1. Contain the special attribute.

AND

2. Identify the local gateway with the other attributes.

Use mapping A. In other cases, use mapping B.

Mapping A

1. Map the domain defined attribute value to ASCII, as defined in chapter 3.

2. Where appropriate (P1 recipients), interpret the string according to the semantics implied by the domain defined attribute.

Mapping B.

This will be used for X.400 addresses which do not use the explicit RFC 822 encoding.

1. Noting the hierarchy specified in 4.3.1, determine the maximum set of attributes which have an associated domain specification. If no match is found, allocate

the domain as the domain specification of the local gateway, and go to step 4.

2. Following the 4.3.1 hierarchy, if each successive component exists, and conforms to the syntax EBNF.domain-syntax (as defined in 4.3.1), allocate the next subdomain.

3. If the remaining components are personal-name components, conforming to the restrictions of 4.2.2, then EBNF.encoded-pn should be derived to form 822.local-part. In other cases the remaining components should simply be encoded as a 822.local-part using the EBNF.std-orname syntax. Where registered domain defined types exist, the DD. syntax should not be used.

4. If this step is reached for an 822-P1 recipient, then the address is invalid. For other addresses, if the derived 822.local-part can only be encoded by use of 822.quoted-string, the gateway may optionally use the ASCII to 822.local-part mapping defined in Appendix A, dependent on the mail protocols of the networks being relayed to. Use of this encoding is discouraged.

4.3. Repeated Mappings

The mappings defined are symmetrical across a single gateway, except in certain pathological cases (see chapter 3). However, it is always possible to specify any valid address across a gateway. This symmetry is particularly useful in cases of (mail exploder type) distribution list expansion. For example, an X.400 user sends to a list on an RFC 822 system which he belongs to. The received message will have the originator and any 3rd party X.400 O/R names in correct format (rather than doubly encoded). In cases (X.400 or RFC 822) where there is common agreement on gateway identification, then this will apply to multiple gateways.

However, the syntax may be used to source route.

For example: X.400 -> RFC 822 -> X.400

```
C               = "UK"
ADMD            = "BT"
PRMD            = "AC"
"JNT-Mail"      = "/PN=Duval/DD.Title=Manager/(a)FR.PTT.Inria"
```

This will be sent to an arbitrary UK Academic Community gateway by X.400. Then by JNT Mail to another gateway determined by the domain FR.PTT.Inria. This will then derive the X.400 O/R Name:

```
C               = "FR"
ADMD            = "PTT"
PRMD            = "Inria"
PN.S            = "Duval"
"Title"         = "Manager"
```

Similarly: RFC 822 -> X.400 -> RFC 822

```
"/C=UK/ADMD=BT/PRMD=AC/RFC-822=jj(a)seismo.css.gov/"
                                                @monet.berkeley.edu

/C=UK/ADMD=BT/PRMD=AC/RFC-822=jj#l#a#r#seismo.css.gov/
                                                @monet.berkeley.edu
```

The second case uses the Appendix A encoding to avoid 822.quoted-text. This will be sent to monet.berkeley.edu by RFC 822, then to the AC PRMD by X.400, and then to jj@seismo.css.gov by RFC 822.

4.4. The full P1 / 822-P1 mapping

There are two basic mappings at the P1 level:

1. 822-P1 return address <-> P1.UMPDUEnvelope.originator

2. 822-P1 recipient <-> P1.RecipientInfo

822-P1 recipients and return addresses are encoded as EBNF.822-address. As P1.UMPDUEnvelope.originator is encoded as P1.ORName, mapping 1) has already been specified. P1.RecipientInfo contains a P1.ORName and additional information. The handling of this additional information is now specified.

4.4.1. RFC 822 -> X.400

The following default settings should be made for each component of P1.RecipientInfo.

P1.ExtensionIdentifier

This can be set systematically by the X.400 system.

P1.RecipientInfo.perRecipientFlag

Responsibility Flag should be set. Report Request should be set according to content return policy, as discussed in section 5.3. User Report Request should be set to Basic.

P1.ExplicitConversion

This optional component should be omitted.

4.4.2. X.400 -> RFC 822

The mapping only takes place in cases where P1.RecipientInfo.perRecipientFlag Responsibility Flag is set. The following treatment should be given to the other P1.RecipientInfo components.

P1.ExtensionIdentifier

Not used.

P1.RecipientInfo.perRecipientFlag

If ReportRequest is Confirmed or Audit-and-Confirmed then a delivery report indicating success should be sent by the gateway. This report should use each P1.ReportedRecipientInfo.SupplementaryInformation to indicate the identity of the gateway, and the nature of the report (i.e. only as far as the gateway). Failures will be handled by returning RFC 822 messages, and so User Report Request set to No report is ignored.

P1.ExplicitConversion

If present, the O/R name should be rejected, unless the requested conversion can be achieved. None of the currently recognised values of this parameter are appropriate to a gateway using this specification.

4.5. The full P2 / RFC 822 mapping

All RFC 822 addresses are assumed to use the 822.mailbox syntax. This should include all 822.comments associated with the lexical tokens of the 822.mailbox. All P2.ORNames are encoded within the syntax P2.ORDescriptor, or P2.Recipient (or within Message IDs). An asymmetrical mapping is defined between these components.

4.5.1. RFC 822 -> X.400

The following sequence is followed.

1. Take the address, and extract an EBNF.822-address. This can be derived trivially from either the 822.addr-spec or 822.route-addr syntax. This is mapped to P2.ORName as described above.

2. A string should be built consisting of (if present):

   - The 822.phrase component if it is a 822.phrase 822.route-addr construct.

   - Any 822.comments, in order, retaining the parentheses.

   This string should then be encoded into T.61 (as described in chapter 3). If the string is not null, it should be assigned to P2.ORDescriptor.freeformName.

3. P2.ORDescriptor.telephoneNumber should be omitted.

4. In cases of converting to P2.Recipient, P2.Recipient.replyRequest and P2.Recipient.reportRequest should be omitted.

If the 822.group construct is present, each included 822.mailbox should be encoded as above. The 822.group should be mapped to T.61, and a P2.ORDesciptor with only a freeformName component built from it.

4.5.2. X.400 -> RFC 822

In the basic case, where P2.ORName is present, proceed as follows.

1. Encode P2.ORName as EBNF.822-address.

2a. If P2.ORDescriptor.freeformName is present, convert it to ASCII (chapter 3), and use use this as the 822.phrase component of 822.mailbox using the 822.phrase 822.route-addr construct.

2b. If P2.ORDescriptor.freeformName is absent, if EBNF.822-address is parsed as 822.addr-spec use this as the encoding of 822.mailbox. If EBNF.822-address is parsed as 822.route 822.addr-spec, then a 822.phrase taken from 822.local-part should be added.

3. If P2.ORDescriptor.telephoneNumber is present, this should be placed in a trailing 822.comment.

4. If P2.Recipient.reportRequest has the receiptNotification bit set, then an 822.comment "(Receipt Notification Requested)" should be appended to the address. The effort of correlating P1 and P2 information is too great to justify the gateway sending Receipt Notifications.

5. If P2.Recipient.replyRequest is present, an 822.comment "(Reply requested)" or "(Reply not requested)" should be appended to the address, dependent on its value.

If P2.ORName is absent, P2.ORDescriptor.freeformName should be converted to ASCII, and used with the RFC 822 822.group syntax:

freeformname ":" ";"

Steps 3-5 should then be followed.

4.6. Message IDs

There is a need to map both ways between 822.msg-id and P2.IPMessageID. A mapping is defined which is symmetrical for non-pathological cases. The mapping allows for the fact that P2.IPMessageID.PrintableString is mandatory for the Cen/Cenelec profile. This allows for good things to happen when messages pass multiple times across the X.400/RFC 822 boundary. A mapping between 822.msg-id and P1.MPDUIdentifier is defined. This allows for X.400 error messages to reference an RFC 822 ID, which is preferable to a gateway generated ID.

4.6.1. P2.IPMessageID -> 822.msg-id

P2.IPMessageID.ORName is used to generate an 822.addr-spec, as defined above. P2.IPMessageID.PrintableString is mapped to ASCII, as defined in chapter 3. This string (if it is present and if the value is not "RFC-822") is appended to the front of the 822.local-part of the 822.msg-id, with "*" as a separator. If no ORName is present, an 822.msg-id of the form "PrintableString*@gateway-domain" is generated.

4.6.2. 822.msg-id -> P2.IPMessageID

822.local-part is parsed as:

```
[ printablestring "*" ] real-local-part
```

If EBNF.printablestring is found, it is mapped to PrintableString, and used as P2.IPMessageID.PrintableString. Otherwise P2.IPMessageID.PrintableString is set to "RFC-822". This arbitrary value allows for conformance to Cen/Cenelec. If EBNF.real-local-part is not present, no P2.IPMessageID.ORName is generated. Otherwise, 822.local-part is replaced with EBNF.real-local-part, and 822.addr-spec is mapped to P2.IPMessageID.ORName as defined above.

4.6.3. 822.msg-id -> P1.MPDUIdentifier

P1.CountryName is assigned to "", P1.AdministrationDomainName to 822.domain (from 822.msg-id) and P1.MPDUIdentifier.IA5String to 822.local-part (from 822.msg-id).

4.6.4. P1.MPDUIdentifier -> 822.msg-id

822.local-part is set to P1.MPDUIdentifier.IA5String, with any CRLF mapped to SPACE. If P1.CountryName is "", 822.domain is set to P1.AdministrationDomainName; Otherwise to P1.AdministrationDomainName ".." P1.CountryName. If there are any specials, the domain literal encoding should be used.

Chapter 5 -- Protocol Elements

This chapter gives detailed mappings for the functions outlined in chapters 1 and 2. It makes extensive use of the notations and mappings defined in chapters 3 and 4. This chapter is structured as follows:

5.1. Basic RFC 822 -> X.400 mappings

5.2. A definition of some new RFC 822 elements, and their mapping to X.400.

5.3 Some special handling associated with Return of Contents.

5.4. X.400 -> RFC 822

5.1. RFC 822 -> X.400

First, the basic functions of an 822-P1 protocol should be mapped as follows:

822-P1 Originator

Mapped to P1.UMPDUEnvelope.originator (see chapter 4).

822-P1 Recipient

Mapped to P1.RecipientInfo (see chapter 4).

The RFC 822 headers are used to generate both a P1.UMPDUEnvelope and a P2.Heading. The IP Message will have either one or two P2.BodyParts which will be type P2.IA5Text with no P2.IA5Text.repertoire component. The last P2.BodyPart will contain the RFC 822 message body. If there are any RFC 822 headers which indicate mapping into the P2.BodyPart, then two P2.BodyParts are generated. If a revised version of P2 allowed for extensible header specification, this would be seen as a preferable mapping. The first body part will start with the line:

RFC-822-Headers:

The rest of this body part will contain all of the headers not otherwise mapped (both 822.field-name and 822.field-body). The order of any such headers should be preserved. Similarly, ordering within P2.Heading and P1.UMPDUEnvelope should reflect ordering within the RFC 822 header. No P1 or P2 optional fields are generated unless specified.

A pro-forma X.400 message is now specified. Some of these defaults may be changed by the values in the RFC 822 message being mapped. The mandatory P1 and P2 components have the following defaults.

P1.MPDUIdentifier

The default should be unique value generated by the gateway.

P1.OriginatorORName

Always generated from 822-P1.

P1.ContentType

P1.ContentType.p2

P1.RecipientInfo

These will always be supplied from 822-P1.

P1.Trace

The last P1.TraceInformation component is generated such that: P1.TraceInformation.GlobalDomainIdentifier is set to the local vaglue. P1.DomainSuppliedInfo.action is set to relayed. P1.DomainSuppliedInfo.arrival is set to the current time. P1.DomainSuppliedInfo.previous may be set if there is anything sensible to set it to.

P2.IPMessageID

The default should be a unique value generated by the gateway.

The following optional parameters should be set:

P1.PerMessageFlag

The P1.PerMessageFlag.contentReturnRequest bit should be set according to the discussion in section 5.3. The P1.PerMessageFlag.alternateRecipientAllowed bit should be set, as it seems desirable to maximise opportunity for (reliable) delivery.

The RFC 822 headings should be mapped as follows:

Received:

Fudged onto P1.TraceInformation (try not to grimace too much). P1.DomainSuppliedInfo.action is set to relayed. P1.DomainSuppliedInfo.arrival is set to the date-time component P1.TraceInformation.GlobalDomainIdentifier has P1.CountryName as a null string, and P1..AdministrationDomainName as the domain of the receiving host (if present - null string if not). P1.DomainSuppliedInfo.previous has P1.CountryName as a null string, and P1.AdministrationDomainName has the domain of the sending host with all other information enclosed in round parentheses. The encoding of ASCII to PrintableString (chapter 3) should be used if needed. For example:

```
Received: from 44e.cs.ucl.ac.uk by vax2.Cs.Ucl.AC.UK
                with SMTP  id a002110; 18 Dec 85 10:40 GMT
```

maps to -

```
P1.GlobalDomainIdentifier
   CountryName                  = ""
   AdministrationDomainName     = "vax2.Cs.Ucl.AC.UK"
P1.DomainSuppliedInfo
   arrival                      = 18 Dec 85 10:40 GMT
   action                       = relayed
   previous
      CountryName               = ""
      AdministrationDomainName  =
             "44e.cs.ucl.ac.uk (with SMTP id a002110)"
```

Date:

This is used to set the first component of P1.TraceInformation. The mandatory components are set as follows:

```
P1.GlobalDomainIdentifier
   CountryName                  = ""
   AdministrationDomainName     = ""
P1.DomainSuppliedInfo
   arrival                      = time derived from Date:
   action                       = relayed
```

No optional fields are used in the trace.

Message-Id:

Mapped to P2.IPMessageID. If the RFC 822 message does not contain a P1-Message-ID: field, the Message-Id: field is also mapped to P1.MPDUIdentifier. For these, and all other fields containing msg-id the mappings of chapter 4 are used for each msg-id.

From:

If Sender: is present, this is mapped to P2.AuthorisingUsers. If not, it is mapped to P2.Originator. For this, and other components containing addresses, the mappings of chapter 4 are used for each address.

Sender:

Mapped to P2.Originator.

Reply-To:

Mapped to P2.Heading.replyToUsers.

To:

Mapped to P2.Heading.primaryRecipients

Cc:

Mapped to P2.Heading.copyRecipients.

Bcc:

Mapped to P2.Heading.blindCopyRecipients.

In-Reply-To:

Mapped to P2.Heading.inReplyTo for the first (if any) 822.msg-id component. If the field contains an 822.phrase component, or there are multiple 822.msg-id components, the ENTIRE field is passed in the P2.BodyPart.

References:

Mapped to P2.Heading.crossReferences.

Keywords:

Passed in the P2.BodyPart.

Subject:

Mapped to P2.Heading.subject. The field-body uses the mapping referenced in chapter 3 from ASCII to T.61.

Comments:

Passed in the P2.BodyPart.

Encrypted:

Passed in the P2.BodyPart.

Resent-*

Passed in the P2..BodyPart <8>.

Other Fields

In particular X-* fields, and "illegal" fields in common usage (e.g. "Fruit-of-the-day:") are passed in the P2.BodyPart. The same treatment should be applied to RFC 822 fields where the content of the field does not conform to RFC 822 (e.g. a Date: field with unparsable syntax).

5.2. Extended RFC 822 Elements -> X.400

First an EBNF definition of a number of extended fields is given, and then a mapping to X.400 is defined. In most cases, the RFC 822 syntax is defined to make this mapping very straightforward, and so no detailed explanation of the mapping is needed.

```
extended-field  = "P1-Message-ID" ":" p1-msg-id
                / "X400-Trace" ":" x400-trace
                / "Original-Encoded-Information-Types"
                   ":"encoded-info
                / "P1-Content-Type" ":" p1-content-type
                / "UA-Content-ID" ":" printablestring
                / "Priority" ":" priority
                / "P1-Recipient" : 1 mailbox
                / "Deferred-Delivery" ":" date-time
```

```
                    / "Bilateral-Info" ":" bilateral-info
                    / "Obsoletes" ":" 1 msg-id
                    / "Expiry-Date" ":" date-time
                    / "Reply-By" ":" date-time
                    / "Importance" ":" importance
                    / "Sensitivity" ":" sensitivity
                    / "Autoforwarded" ":" boolean

      p1-msg-id     = global-id ";" *text

      p1-content-type = "P2" / atom

      x400-trace    = global-id ";"
                    "arrival" date-time
                    [ "deferred" date-time ]
                    [ "action" action ]
                    [ "converted" "(" encoded-info ")" ]
                    [ "previous" global-id ]

      action        = "Relayed" / "Rerouted" / escape

      global-id     = c "*" admd [ "*" prmd ]

      encoded-info  = 1 encoded-type

      encoded-type  = "Undefined"          ; undefined (0)
                    / "Telex"              ; tLX (1)
                    / "IA5-Text"           ; iA5Text (2)
                    / "G3-Fax"             ; g3Fax (3)
                    / "TIF0"               ; tIF0 (4)
                    / "Teletex"            ; tTX (5)
                    / "Videotex"           ; videotex (6)
                    / "Voice"              ; voice (7)
                    / "SFD"                ; sFD (8)
                    / "TIF1"               ; tIF1 (9)
                    / escape

      priority      = "normal" / "non-urgent" / "urgent" / escape

      bilateral-info = c "*" admd "*" *text

      importance    = "low" / "normal" / "high" / escape

      sensitivity   = "Personal" / "Private"
                    / "Company-Confidential" / escape

      escape        = 1*DIGIT
```

With the exception of "Bilateral-Info:" and "X400-Trace:", there must be no more than one of each of these fields in an RFC 822 header. Any field beginning with the String "Autoforwarded-" is valid if the field would be syntactically valid with this string removed.

The mappings to X.400 are as follows:

P1-Message-ID:

Mapped to P1.UMPDUEnvelope.MPDUIdentifier. This take precedence over any value derived from Message-ID:.

X400-Trace:

Mapped to the next component of P1.UMPDUEnvelope.Traceinformation. Care should be taken to preserve order. If one or more of these mappings is made, then a trace component should NOT be generated from the Date: field which should be redundant. This is because the message has previously come from X.400, and the Date: information will be redundant. Note that all trace information (Received: and "X400-Trace:") in the RFC 822 message will be in strict order, with the most recent at the top. This order should be preserved in the mapping.

Original-Encoded-Information-Types:

This is used to set P1.UMPDUEnvelope.original. P1.EncodedInformationTypes.[0] has bits set according to each of the encoded-info components in this field. Any escape values should not be encoded.

P1-Content-Type:

If the value is anything other than "P2", the mapping should not be performed (unless the new value has some semantics to the gateway).

UA-Content-ID:

Mapped to P1.UMPDUEnvelope.UAContentID.

Priority:

Mapped to P1.UMPDUEnvelope.Priority. An escape value should be encoded as P1.Priority.normal.

P1-Recipient:

If this field is set, the P1.PerMessageFlag.discloseRecipients bit should be set. Any of the addresses here which do not correspond to 822-P1 recipients should be added to the P1 recipient list, with the responsibility bit turned off.

Deferred-Delivery:

Mapped to P1.UMPDUEnvelope.deferredDelivery. Note that the value of this field should always be in the past, as this field should only be present in messages which have come originally from X.400. Thus there should be no implied action. See also the comments on the reverse mapping.

Bilateral-Info:

No attempt is made to reconvert this information back to X.400.

Obsoletes:

Mapped to P2.Heading.obsoletes.

Expiry-Date:

Mapped to P2.Heading.expiryDate.

Reply-By:

Mapped to P2.Heading.replyBy.

Importance:

Mapped to P2.Heading.importance. An escape value should be encoded as P2.Heading.importance.normal.

Sensitivity:

Mapped to P2.Heading.sensitivity. An escape value should be encoded as P2.Heading.sensitivity.normal.

Autoforwarded:

If this field is present and the value is "TRUE", there will be zero or more field names beginning "Autoforwarded-".

These should be taken, and the string "Autoforwarded-" stripped. These fields, in conjunction with the 822-P1 information should be used to build an IP Message. Any implied actions should be taken. P2.Heading.autoforwarded is set in this message. The other RFC 822 fields are used to build another IP Message, which is used as the single body part of the first message. This mechanism does not nest.

5.3. Return of Contents

It is not clear how widely supported X.400 return of contents service will be. However, profiling work suggests that most systems will not support this service. As this service is expected in the RFC 822 world, two approaches are specified (it is not so necessary in the X.400 world, as delivery reports are distinguished from messages). The choice will depend on the service level of the X.400 community being serviced by the gateway.

In environments where return of contents is widely supported, the P1.PerMessageFlag content return request bit will be set, and the Report Request bit in P1.PerRecipientFlag will be set to Confirmed, for every message passing from RFC 822 -> X.400. The content return service can then be passed back to the end (RFC 822) user in a straightforward manner.

In environments where return of contents is not widely supported, a gateway must make special provisions to handle return of contents. For every message passing from RFC 822 -> X.400, the P1.PerMessageFlag content return request bit will be set, and the Report Request bit in P1.PerRecipientFlag will be set to Confirmed. When the delivery report comes back, the gateway can note that the message has been delivered to the recipient(s) in question. If a non-delivery report is received, a meaningful report (containing some or all of the original message) can be sent to the 822-P1 originator. If no report is received for a recipient, a (timeout) failure notice should be sent to the 822-P1 originator. The gateway may retransmit the X.400 message if it wishes. Delivery confirmations should only be sent back to the 822-P1 originator if the P1.PerRecipientFlag User Report Request bit is set to Confirmed.

5.4. X.400 -> RFC 822

5.4.1. General

This section describes how to build a pro-forma message, and then explains how these defaults may be overridden. It should be noted that RFC 822 folding of headers should be used in an appropriate manner.

5.4.2. Service MPDU

5.4.2.1. Probe

Any P1.ProbeMPDU should be serviced by the gateway, as there is no equivalent RFC 822 functionality. The value of the reply is dependent on whether the gateway could service a User MPDU with the values specified in the probe. The reply should make use of P1.SupplementaryInformation to indicate that the probe was serviced by the gateway.

5.4.2.2. Delivery Report

The 822-P1 components are constructed as follows:

822-P1 Originator

This is set to an 822.addr-spec pointing to an administrator at the gateway.

822-P1 Recipient

The single recipient is constructed from P1.DeliveryReportEnvelope.originator, using the mappings of chapter 4.

The mandatory RFC 822 headers for an RFC 822 pro-forma are constructed as follows:

Date:

From the P1.DomainSuppliedInfo.arrival component of the first P1.TraceInformation component.

From:

This is set to the same as the 822-P1 originator. An

appropriate phrase component may be added (e.g. giving the name of the gateway).

To:

The same as the 822-P1 recipient.

A Subject: field should be added, which contains some appropriate words (e.g. "Delivery Report").

The other two P1.DeliveryReportEnvelope parameters should be mapped as follows:

P1.DeliveryReportEnvelope.report

This should be mapped to a P1-Message-Id: field.

P1.DeliveryReportEnvelope.TraceInformation

Each component should be mapped to an "X400-Trace:" field. RFC 822 and X.400 ordering should be maintained (see 5.3).

The P1.DeliveryReportContent parameters should be mapped to a series of new RFC 822 headers. These new headers are intended for processing in the RFC 822 world. No attempt will be made to reverse the mappings.

```
drc-field    = "Delivery-Report-Content-Original"
               ":" msg-id
  / "Delivery-Report-Content-Intermediate-Trace"
               ":" x400-trace
  / "Delivery-Report-Content-UA-Content-ID"
               ":" printablestring
  / "Delivery-Report-Content-Billing-Information"
               ":" *text
  / "Delivery-Report-Content-Reported-Recipient-Info"
               ":" drc-info

drc-info     = mailbox ";"
               last-trace ";"
               "ext" 1*DIGIT
               "flags" 2DIGIT
               [ "intended" mailbox ] ";"
               [ "info" printablestring ]
```

```
last-trace   = drc-report ";"
               date-time ";"
               [ "converted" "(" encoded-info ")"

drc-report   = "SUCCESS" drc-success
               / "FAILURE" drc-failure

drc-success  = date-time ";" 1*DIGIT

drc-failure  = *text [ ";" *text ] ";"
```

There may be multiple "Delivery-Report-Content-Intermediate-Trace:" and "Delivery-Report-Content-Reported-Recipient-Info:" fields. The msg-id for "Delivery-Report-Content-Original" is derived according to the mapping of chapter 4. EBNF.drc-failure may use numeric values or textual explanation. The latter is preferred. All P1.DeliveryReportContent parameters are mapped to the corresponding component. The order of "Delivery-Report-Content-Intermediate-Trace:" should have the most recently stamped one first.

The body of the RFC 822 message should be a human readable description of the critical parts of the P1.DeliveryReportContent. In particular, the failed recipients, and failure reason should be given. Some or all of the original message should be included in the delivery report. The original message will be available at the gateway, as discussed in section 5.3.

5.4.3. User MPDU

These elements are the basis for both Status Report and IP Message.

The 822-P1 components are constructed as follows:

822-P1 Originator

This is derived from P1.UMPDUEnvelope.originator.

822-P1 Recipient

Each recipient is constructed from the P1.RecipientInfo, as described in chapter 4. This describes actions as well as format mappings.

The mandatory RFC 822 field pro-forma is derived as follows. In most cases where the P1.UMPDUContent is an IP Message, these defaults will be overridden:

Date:

From the P1.DomainSuppliedInfo.arrival component of the first P1.TraceInformation component.

From:

From the P1.UMPDUEnvelope.originator, as defined in chapter 4.

To:

This default is only required if the generated RFC 822 message has no destination specification. If P1.PerMessageFlag.discloseRecipients is set then it should contain the ORName in each P1.RecipientInfo component. If it is not set, the it should be set to "To: No Recipients Specified : ;".

The mappings, and any actions for each P1.UserMPDU element is now considered.

P1.MPDUIdentifier

Mapped to the extended RFC 822 field "P1-Message-ID:". Note that the sequence CRLF is mapped to SPACE, which makes the mapping irreversible for such cases.

P1.UMPDUEnvelope.original

Mapped to the extended RFC 822 field "Original-Encoded-Information-Types:". If it contains only P2.IA5Text, the RFC 822 field may be omitted.

P1.ContentType

As this can currently only have one value, it is not mapped, on the basis that it is redundant. If the field contains any value other than P2, then the UMPDU should be rejected.

P1.UAContentID

Mapped to the extended RFC 822 field "UA-Content-Id:".

P1.Priority

Mapped to the extended RFC 822 field "Priority:".

P1.PerMesageFlag

This has a number of components:

- discloseRecipients

  If this bit is set, a "P1-Recipient:" field should be generated, and contain each of the P1 recipients.

- conversionProhibited

  If this bit is set, the message should be rejected if it contains P2.BodyPart which is not P2.IA5Text or P2.ForwardedIPMessage.

- alternateRecipientAllowed

  The value of this bit is ignored.

- contentReturnRequest

  The value of this bit is ignored.

P1.UMPDUEnvelope.deferredDelivery

This should be mapped to the extended RFC 822 field "Deferred-Delivery:". X.400 profiles, and in particular the CEN/CENELEC profile [CEN/CENELEC/85a], specify that this element must be supported at the first MTA. Thus, it is expected that the value of this element will always be in the past. If it is not, the function may optionally be implemented by the gateway: that is, the gateway should hold the message until the time specified in the protocol element. Thus the extended RFC 822 field is just for information.

P1.PerDomainBilateralInformation

Each component should be encoded in the extended RFC 822 field "Bilateral-Info:". P1.BilateralInfo should be mapped into ASCII in a manner appropriate to its contents. This submapping is not reversible.

P1.TraceInformation

This should be mapped to "X400-Trace:", as for the delivery report.

5.4.4. Status Report

The entire status report is mapped into the body of the RFC 822 message, in the same manner as for a Delivery Report. An appropriate "Subject:" field should be generated. As status reports cannot be requested from the RFC 822 world, the mapping is not likely to be used a great deal.

5.4.5. IP Message

The P1.UMPDUEnvelope pro-forma specification ensures all the 822-P1 information, and a minimal (legal) RFC 822 message. The mappings and actions for the P2.Heading components are now described. Basically, these are interpreted as actions and/or mappings into RFC 822 fields. The following mappings are specified:

P2.IPMessageID

This is mapped to the field "Message-ID:", according to section 4.

P2.Heading.originator

If P2.Heading.authorisingUsers is present this is mapped to Sender:, if not to From:.

P2.Heading.authorisingUsers

Mapped to From:.

P2.Heading.primaryRecipients

Mapped to To:.

P2.Heading.copyRecipients

Mapped to Cc:.

P2.Heading.blindCopyRecipients

Mapped to Bcc:.

P2.Heading.inReplyTo

Mapped to In-Reply-To:.

P2.Heading.obsoletes

Mapped to the extended RFC 822 field "Obsoletes:"

P2.Heading.crossReferences

Mapped to References:.

P2.Heading.subject

Mapped to subject. The contents are converted to ASCII (as defined in chapter 3). Any CRLF are not mapped, but are used as points at which the subject field must be folded. line.

P2.Heading.expiryDate

Mapped to the extended RFC 822 field "Expiry-Date:".

P2.Heading.replyBy

Mapped to the extended RFC 822 field "Reply-By:".

P2.Heading.replyToUsers

Mapped to Reply-To:.

P2.Heading.importance

Mapped to the extended RFC 822 field "Importance:".

P2.Heading.sensitivity

Mapped to the extended RFC 822 field "Sensitivity:".

P2.Heading.autoforwarded

If it is set to FALSE, it is simply mapped to the extended RFC 822 field "Autoforwarded:". If this is set to TRUE, the P2.Body does not consist of a single P2.ForwardedIPMessage, then there is an X.400 error, and the message should be bounced. Otherwise the following steps are taken.

1. The mappings for all of the message, except the body part are completed.

2. Prepend each RFC 822 fieldname with the string "Autoforwarded-". Mapped to the extended RFC 822 field "Autoforwarded:".

3. Add the field "Autoforwarded:" with value TRUE.

4. Convert the syntax of the P2.ForwardedIPMessage to generate the remaining RFC 822 fields.

The P2.Body is mapped into the RFC 822 message body. Each P2.BodyPart is converted to ASCII. If the P2.Body contains a P2.BodyPart not listed here, the entire message should be rejected. If there are exactly two P2.IA5Text body parts, and the first line of the first is "RFC-822-Headers:", then the rest of this first body part should be treated as additional header information for the RFC 822 message. If there is an "In-Reply-To:" field, this should be used to replace any generated In-Reply-To: field.

In other cases of multiple P2.BodyPart, the mapping defined by Rose and Stefferud in [Rose85b], should be used to separate the P2.BodyParts in the single RFC 822 message body.

Individual body parts are mapped as follows:

P2.IA5Text

The mapping is trivial.

P2.TTX

If any P1.Teletex.NonBasicParams are set, the message should be rejected. Otherwise, it should be converted to ASCII according to chapter 3.

P2.SFD

An SFD should be converted to ASCII as if it was being rendered on an 79 column ASCII only VDU. It seems likely that many gateways will not support this conversion. In these cases, the message should be rejected.

P2.ForwardedIPMessage

The P2.ForwardedIPMessage.delivery and P2.ForwardedIPMessage.DeliveryInformation are discarded <9>. The IM-UAPDU should have its syntax mapped (recursively) according to this gateweying specification. Clearly, it makes no sense to apply any of the actions defined here.

Appendix A -- Quoted String Encodings

This Appendix describes a quoting mechanism which may be used to allow general interworking between RFC 822, and variants of RFC 822 which do not support 822.quoted-string. This is important, as the basic X.400 <-> RFC 822 mapping makes use of 822.quoted-string.

1. ASCII <-> 822.atom

The following EBNF is specified.

```
atom-encoded    = *( a-char / a-encoded-char )
a-char          = <any CHAR except specials, SPACE,
                        CTL, "_", and "#">
a-encoded-char  = "_"                    ; (space)
                / "#u#"                  ; (_)
                / "#l#"                  ; <(>
                / "#r#"                  ; <)>
                / "#m#"                  ; (,)
                / "#c#"                  ; (:)
                / "#b#"                  ; (\)
                / "#h#"                  ; (#)
                / "#e#"                  ; ($=)
                / "#s#"                  ; ($/)
                / "#" 3DIGIT "#"
```

NOTE: There are two encodings of double characters. This is so that systems using this encoding, do not also need to know about the "$" quoting mechanism defined in chapter 4.

The 822.3DIGIT in EBNF.a-encoded-char must have range 0-127 (Decimal), and is interpreted in decimal as the corresponding ASCII character. The choice of special abbreviations (as opposed to octal encoding) provided is based on the manner in which this mapping is most frequently used: encoding PrintableString components of O/R names as atom. Therefore, there are special encodings for each of the PrintableString characters not in EBNF.a-char, except ".". Space is given a single character encoding, due to its (expected) frequency of use, and backslash as the RFC 822 single quote character.

To encode (ASCII -> atom): all EBNF.a-char are used directly and all other CHAR are encoded as EBNF.a-encoded-char. To decode (822.atom -> ASCII): if 822.atom can be parsed as EBNF.encoded-atom reverse the previous mapping. If it cannot be so parsed, map the characters directly.

2. 822.local-part <-> ASCII

A related transformation is for 822.local-part (or other element defined as '822.word ("." 822.word)') where not 822.quoted-text is used. To encode (ASCII -> 822.local-part), all EBNF.a-char and "." are used directly and all other 822.CHAR are encoded as EBNF.a-encoded-char. To decode (822.local-part -> ASCII), first attempt to parse 822.local-part as '822.atom *("." 822.atom)'. If this fails, or if each 822.atom cannot be parsed as EBNF.atom-encoded then map each character directly. Otherwise map each "." directly, and each atom as in the previous section.

There are places where it is needed to convert between PrintableString or IA5Text (X.400), and 822.word (RFC 822). word may be encoded as 822.atom (which has a restricted character set) or as 822.quoted-string, which can handle all ASCII characters. If 822.quoted-string is used, clearly the mappings for PrintableString defined in Chapter 3 provide a straightforward mapping. However, some RFC 822 based networks cannot handle the 822.quoted-string format in all cases. This Appendix is for use in these cases. The major requirement for this mapping is the UNIX world, but it may well be needed in other places.

These mappings are somewhat artificial, and their usage is discouraged, except in cases where there is no alternative.

Appendix B -- Mappings Specific to JNT Mail

This Appendix is specific to the JNT Mail Protocol. It describes specific changes in the context of this protocol. Addressing is not discussed here, as it is covered in Appendix A.

1. Introduction

There are four aspects of a gateway which are JNT Mail Specific, in addition to those relating to addressing. These are each given a section of this appendix.

2. Acknowledge-To:

This field has no direct functional equivalent in X.400. However, it can be supported to an extent, and can be used to improve X.400 support.

When going from JNT Mail to X.400, the User Report Request bits of each P1.RecipientInfo.perRecipientFlag should be set to confirmed. If there is more that one address in the Acknowledge-To: field, or if the one address is not equivalent to the 822-P1 return address, then:

a. Acknowledgement(s) should be generated by the gateway. The text of these acknowledgements should indicate that they are generated by the gateway.

b. The Acknowledge-To: field should also be passed in the first P2.BodyPart.

When going from X.400 to JNT Mail, in cases where P1.RecipientInfo.perRecipientFlag has the user bits set to confirmed the copy of the message to that recipient should have an Acknowledge-To: field containing the P.UMPDUEnvelope.originator. No attempt should be made to map Receipt notification requests onto Acknowledge-To:. This is because no association can be guaranteed between P2 and P1 level addressing information.

3. Trace

JNT Mail trace uses the Via: syntax. When going from JNT Mail to X.400, the following mapping onto P1.TraceInformation is used.

P1.DomainSuppliedInfo.action is set to relayed.

P1.DomainSuppliedInfo.arrival is set to the date-time component

of the Via: field. P1.DomainSuppliedInfo.previous has P1.CountryName as a null string, and P1.AdministrationDomainName as the domain specified in the Via: field.
P1.TraceInformation.GlobalDomainIdentifier has P1.CountryName as a null string, and P1.AdministrationDomainName as any commented information in the Via: field. For example:

```
Via: UK.AC.Edinburgh ; 17 Jun 85 9:15:29 BST (EMAS V7)

maps to -

P1.GlobalDomainIdentifier
   CountryName                  = ""
   AdministrationDomainName     = "(EMAS V7)"
P1.DomainSuppliedInfo
   arrival                      = 17 Jun 85 9:15:29 BST
   action                       = relayed
   previous
      CountryName               = ""
      AdministrationDomainName  = "UK.AC.Edinburgh"
```

4. Timezone specification

The extended syntax of zone defined in the JNT Mail Protocol should be used in the mapping of UTCTime defined in chapter 3.

5. Lack of separate 822-P1 originator specification

In JNT Mail the default mapping of the P1.MPDUEnvelope.originator is to the Sender: field. This can cause a problem if the mapping of P2.Heading has already generated a Sender: field. To overcome this, new extended JNT Mail field is defined. This is chosen to align with the JNT recommendation for interworking with full RFC 822 systems [Kille84b].

```
original-sender     = "Original-Sender" ":" mailbox
```

If an IPM has no P2.heading.authorisingUsers component and P2.Heading.originator.ORName is different from P1.UMPDUEnvelope.originator, map P1.MPDUEnvelope.originator onto the Sender: field.

If an IPM has a P2.heading.authorisingUsers component, and P2.Heading.originator.ORName is different from P1.UMPDUEnvelope.originator, P1.MPDUEnvelpoe.originator should be

mapped onto the Sender: field, and P2.Heading.originator mapped onto the Original-Sender: field.

In other cases the P1.MPDUEnvelope.Originator is already correctly represented.

Note that in some pathological cases, this mapping is asymmetrical.

Appendix C -- Mappings Specific to Internet Mail

The Simple Mail Transfer Protocol [Postel82a] is used in the ARPA-Internet, and in any network following the US DoD standards for internetwork protocols. This appendix is specific to those hosts which use SMTP to exchange mail.

1. Mapping between O/R names and SMTP addresses

   The mappings of Chapter 4 are to be used.

2. Use of the ARPA Domain System

   Whenever possible, domain-qualified addresses should be be used to specify encoded O/R names. These domain encodings naturally should be independent of any routing information.

3. Identification of gateways

   The ARPA-Internet Network Information Center (NIC) will maintain a list of registered X.400 gateways in the ARPA Internet.

Appendix D -- Mappings Specific to Phonenet Mail

There are currently no mappings specific to Phonenet Mail.

Appendix E -- Mappings Specific to UUCP Mail

Gatewaying of UUCP and X.400 is handled by first gatewaying the UUCP address into RFC 822 syntax (using RFC 976) [Horton86a] and then gatewaying the resulting RFC 822 address into X.400. For example, an X.400 address

Country	US
Organization	Xerox
Personal Name	John Smith

might be expressed from UUCP as

```
inthop!gate!gatehost.COM!/C=US/O=Xerox/PN=John.Smith/
```

(assuming gate is a UUCP-ARPA gateway and gatehost.COM is an ARPA-X.400 gateway) or

```
inthop!gate!Xerox.COM!John.Smith
```

(assuming that Xerox.COM and /C=US/O=Xerox/ are equivalent.)

In the other direction, a UUCP address Smith@ATT.COM, integrated into 822, would be handled as any other 822 address. A non-integrated address such as inthop!dest!user might be handled thought a pair of gateways:

Country	US
ADMD	ATT
PRMD	ARPA
Organization	GateOrg
RFC-822	inthop!dest!user@gatehost.COM

or through a single X.400 to UUCP gateway:

Country	US
ADMD	ATT
PRMD	UUCP
Organization	GateOrg
UUCP	inthop!dest!user

Appendix F -- Format of Address Mapping Tables

There is a need to specify the association between the domain and X.400 namespaces described in 4.2.1. This is defined as a table syntax, but the syntax is defined in a manner which makes it suitable for use with domain nameservers (such as the DARPA Domain nameservers or the UK NRS). The symmetry of the mapping is not clear, so a separate table is specified for each direction. For domain -> X.400:

domain-syntax "#" dmn-orname "#"

For example:

AC.UK#PRMD$DES.ADMD$BT.C$UK#
XEROX.COM#O$Xerox.ADMD$ATT.C$US#

For X.400 -> domain:

dmn-orname "#" domain-syntax "#"

EBNF.domain-syntax will be interpreted according to RFC 920. EBNF.dmn-orname will have components ordered as defined in section 4.2.1, and with the most significant component on the RHS.

References

Bonacker85a.

K.H. Bonacker, U. Pankoke-Babatz, and H. Santo, "EAN - Conformity to X.400 and DFN-Pflichtenheft," GMD (Gesellschaft fur Mathematik und Datenverarbeitung) report, June 1985.

CCITT84a.

CCITT SG 5/VII, "Recommendations X.400," Message Handling Systems: System Model - Service Elements, October 1984.

CCITT84b.

CCITT SG 5/VII, "Recommendations X.411," Message Handling Systems: Message Transfer Layer, October 1984.

CCITT84c.

CCITT SG 5/VII, "Recommendations X.420," Message Handling Systems: Interpersonal Messaging User Agent Layer, October 1984.

CCITT84d.

CCITT SG 5/VII, "Recommendations X.409," Message Handling Systems: Presentation Transfer Syntax and Notation, October 1984.

CEN/CENELEC/85a.

CEN/CENELEC/Information Technology/Working Group on Private Message Handling Systems, "FUNCTIONAL STANDARD A/3222," CEN/CLC/IT/WG/PMHS N 17, October 1985.

Crocker82a.

D.H. Crocker, "Standard of the Format of ARPA Internet Text Messages," RFC 822, August 1982.

Horton85a.

M.R. Horton, "Draft Standard for ARPA/MHS Addressing Gateways," AT&T Bell Laboratories, October 1985.

Horton86a.

M.R. Horton, "UUCP Mail Interchange Format Standard", RFC 976, February 1986.

ICL84a.

ICL, "Comparison of service elements of Grey Book Mail and X.400," Mailgroup Note 18: Extract from unpublished report for ITSU (Information Technology Standards Unit), July 1984.

Kille84a.

S.E. Kille, (editor), JNT Mail Protocol (revision 1.0), Joint Network Team, Rutherford Appleton Laboratory, March 1984.

Kille84b.

S.E. Kille, "Gatewaying between RFC 822 and JNT Mail," JNT Mailgroup Note 15, May 1984.

Kille86a.

S.E. Kille, "O/R Names in the UK Academic Community," UK Working Document, March 1986.

Larmouth83a.

J. Larmouth, "JNT Name Registration Technical Guide," Salford University Computer Centre, April 1983.

Neufeld85a.

G. Neufeld, J. Demco, B. Hilpert, and R. Sample, "EAN: an X.400 message system," in Second International Symposium on Computer Message Systems, Washington, pp. 1-13, North Holland, September 1985.

Postel82a.

J. Postel, "Simple Mail Transfer Protocol," RFC 821, August 1982.

Postel84a.

J. Postel and J. Reynolds, "Domain Requirements," RFC 920, October 1984.

Rose85a.

M.T. Rose, "Mapping Service Elements between ARPA and MHS," Draft proposal, October 1985.

Rose85b.

M.T. Rose and E.A. Stefferud, "Proposed Standard for Message Encapsulation," RFC 934, January 1985.

Notes:

<0> UNIX is a trademark of Bell Laboratories.

<1> The term gateway is used to describe a component performing the protocol mappings between RFC 822 and X.400. This is standard usage amongst mail implementors, but should be noted carefully by transport and network service implementors. (Sometime called a "mail relay".)

<2> If the remote protocol is JNT Mail, a notification may also be sent by the recipient UA.

<3> The asymmetry occurs where an ASCII string contains the sequence EBNF.ps-encoded-char. This would be mapped directly to PrintableString, but the reverse mapping would be to the value implied by the sequence.

<4> It might be suggested that for reasons of elegance, EBNF.ps-delim (left parenthesis) is encoded as EBNF.ps-encoded-char. This is not done, as it it would never be possible to represent a PrintableString containing the character "(" in ASCII. This is because an "(" in ASCII would be mapped to the encoding in PrintableString.

<5> In practice, a gateway will need to parse various illegal variants on 822.date-time. In cases where 822.date-time cannot be parsed, it is recommended that the derived UTCTime is set to the value at the time of translation.

<6> P2.ORname is defined as P1.ORName.

<7> This recommendation may change in the light of CCITT or CEN/CENELEC guidelines on the use of initials.

<8> It would be possible to use a ForwardedIPMessage for these fields, but the semantics are (arguably) slightly different, and it is probably not worth the effort.

<9> Although this violates chapter 1, part 4, principles 2 and 3, it is suggested that this is justified by principle 1.