MASTER EN SCIENCES INFORMATIQUES

Interconnexion entre systèmes de courrier électronique : spécification d'une passerelle et implémentation de la correspondance des adresses

Degueldre, Pascal; Fauvarque, Vincent

Award date:
1988

Awarding institution:
Universite de Namur

Interconnexion entre systèmes de courrier électronique : spécification d'une passerelle et implémentation de la correspondance des adresses.

Mémoire présenté par

Pascal Degueldre

et

Vincent Fauvarque

en vue de l'obtention du titre de

LICENCIE ET MAITRE EN INFORMATIQUE

Promoteur du mémoire :

Ph. VAN BASTELAER

Année Académique 1987-1988

RESUME :

Le sujet de ce mémoire concerne l'interconnexion entre systèmes de courrier électronique. Une passerelle entre des systèmes respectant d'un côté les protocoles X400 et de l'autre le protocole RFC-822 y est spécifiée. Cette spécification est basée sur le document RFC-987 écrit par Steve Kille. La correspondance des adresses de type RFC-822 en format X400 est implémentée.

ABSTRACT :

The subject of this dissertation is the interworking of electronic mail systems. A gateway between the two protocols worlds which are on the one hand the RFC-822 protocol and the other hand X400 protocols is specified. This spécification is based on the RFC-987 document, written by Steve Kille. Only the mapping of addresses from RFC-822 to X400 is implemented.

TOME I

Nous tenons à remercier Monsieur Philippe van Bastelaer d'avoir accepté la direction de ce mémoire et proposé ce sujet intéressant.

Nous tenons également à adresser nos remerciements à Monsieur Patrick Geurts. Qu'il veuille bien trouver, ici, l'expression de notre reconnaissance pour ses conseils et sa grande disponibilité sans lesquels ce texte n'aurait pu voir le jour.

Notre reconnaissance se porte également à tous ceux qui, de près ou de loin, nous ont aidés à mener à bien ce travail et tout particulièrement à Marie-Elise et Isabelle pour les efforts qu'elles ont consentis pour rendre agréable la présentation de ce mémoire.

LISTE DES ABREVIATIONS.

ADMD = ADministration Management Domain.

BNF = Backus Naur Form.

CCITT = Comité Consultatif International de Télégraphie et de téléphonie.

CRLF = Carriage Return/Line Feed.

IM-UAPDU = Interpersonal Message User Agent Protocol Data Unit.

IPM-UAE = Interpersonal Messaging User Agent Entity.

IP-message = Interpersonal message.

IPMS = Interpersonal Messaging Service.

ISO = International Standard Organization.

MD = Management Domain.

MHS = Message Handling System.

MPDU = Message Protocol Data Unit.

MTAE = Message Transfer Agent Entity.

MTL = Message Transfer Layer.

MTS = Message Transfer System.

NIC = Network Information Center.

OPDU = Operation Protocol Data Unit.

ORName = Originator or Recipient Name.

OSI = Open Systems interconnection.

PDU = Protocol Data Unit.

PRMD = PRivate Management Domain.

RARE = Réseaux Associés pour la Recherche Européenne.

SMPDU = Service Message Protocol Data Unit.

SMTP = Simple Mail Transfer Protocol.

SR-UAPDU = Status Report User Agent Protocol Data Unit.

UAE = User Agent Entity.

UAL = User Agent Layer.

UAPDU = User Agent Protocol Data Unit.

UMPDU = User Message Protocol Data Unit.

LISTE DES NORMES.

RFC-821 : Simple Mail Transfer Protocol.

RFC-822 : Standard for the format Internet Text Messages.

RFC-920 : Domain Requirements.

RFC-987 : Mapping between X400 and RFC-822.

X400 : Data Communication Networks Message Handling System

X409 : Presentation Transfer Syntax and Notation

X411 : Message Transfer Layer

X420 : Interpersonal Messaging User Agent Layer

TABLE DES MATIERES.

CHAPITRE I : INTRODUCTION.

Notre époque est profondément marquée par le progrès de l'information : ordinateur, télévision, télécommunications, ont en quelques années envahi notre univers économique, social et culturel. Ces innovations technologiques conduisent à l'apparition de systèmes électroniques de communications qui pourraient marquer la fin de "l'ère de Gutenberg". Cette prolifération de nouveaux moyens de communications a donné naissance à la messagerie électronique qui offre une distibution rapide et simple de messages et un accès à plus d'informations.

Le réseau américain Internet a compris le besoin indispensable d'établir des normes pour assurer la compatibilité des échanges de messages dans son système de courrier électronique. Le résultat de cet effort fut la syntaxe RFC-822. Peu à peu, le succès toujours grandissant de ces services de transmission de messages, ainsi que l'impossibilité de communication entre deux systèmes différents, ont conduit à une normalisation pour l'établissement d'un système de messagerie mondial. C'est pourquoi, le CCITT a publié les Recommandations X400 qui proposent des protocoles standards de communication entre les systèmes de traitement de messages.

La première étape de l'évolution vers un monde se référant totalement à X400 est l'établissement d'une passerelle . Dans ce cadre, le document RFC-987, élaboré par Steve Kille, propose une réalisation d'interconnexion entre ces deux mondes de protocole; il se subdivise en la correspondance des adresses et la correspondance des services.

Notre travail s'est attaché à analyser ce problème et à implémenter la correspondance d'une adresse de type RFC-822 en format X400.

Ce texte s'articule selon le schéma suivant : après une brève présentation des fonctionnalités d'une passerelle et des problèmes sous-jacents (chapitre II), le chapitre III décrit la structure générale du problème . Par souci de modularisation, la correspondance des adresses (chapitre IV) est détaillée indépendamment des éléments de service et de protocole. Elle se présente dès lors comme une "boîte noire" pour la correspondance des services (chapitre V).

Le chapitre VI se rapporte à la réalisation d'un logiciel de correspondance d'adresses de type RFC-822 en format X400 et le chapitre VII résume les axes essentiels du travail.

CHAPITRE II : LES SYSTEMES DE COURRIER ELECTRONIQUE ET LES PASSERELLES.

La téléinformatique est le résultat de la fusion de deux progrès technologiques : les télécommunications et l'informatique. L'objectif de la téléinformatique est de transmettre à distance, par des voies de communication, des informations traitées au départ et à l'arrivée par l'informatique. Cette évolution et l'implantation de vastes réseaux de transmission de données a profondément influencé l'organisation des services informatiques. Le centre informatique se voit remplacé par un réseau informatique i.e. un ensemble d'ordinateurs dispersés, connectés entre eux et bénéficiant d'une plus grande autonomie. L'exploitation des réseaux a imposé un développement important d'applications telles que le transfert de fichiers et la messagerie électronique.

Ce développement du traitement de l'information des données a nécessité la normalisation des applications téléinformatiques. C'est pourquoi des organismes tels que le CCITT (Comité Consultatif International de Télégraphie et de Téléphonie) et l'ISO (International Standard Organization) ont été institués au niveau mondial. Le but des communautés internationales est de fournir un ensemble très riche de spécifications de telle sorte que des organisations variées puissent satisfaire leurs besoins en communication de données respectant les protocoles standards. Suivant les exigences de la communication, une organisation choisit un sous-ensemble des règles et des options offertes.

Actuellement, il existe beaucoup de systèmes de courrier électronique qui respectent des normes différentes. L'interconnexion entre deux messageries distinctes est établie par l'implémentation d'une passerelle. Mais la maintenance de plusieurs courriers posent beaucoup de problèmes dus aux passerelles; d'où l'adoption des normes internationales X400 est une stratégie pour réduire ces problèmes et améliorer les services à long terme.

La fonction de base d'une passerelle est d'échanger des messages entre deux courriers électroniques respectant des protocoles différents. Mais l'interconnexion n'offre pas toute la transparence souhaitée et ne fournit pas tous les services des systèmes de courrier. Un message peut être considéré comme étant composé d'une enveloppe, d'un en-tête et d'un corps. L'enveloppe contient les informations de routage telles que l'adresse de l'expéditeur et du destinataire. L'en-tête est constitué de champs qui donnent une indication des fonctionnalités du système de courrier électronique puisqu'ils sont utilisés pour fournir les services à l'utilisateur. Le corps contient les informations que l'utilisateur souhaitent échanger. La passerelle doit être capable de reconnaître et de créer les messages, dans le format approprié de l'enveloppe, de l'en-tête et du corps des messageries électroniques qu'elle interconnecte. Cinq rôles sont joués par une passerelle :

I) l'établissement d'un interface avec les systèmes de courrier.

Il doit exister un mécanisme tel que les messages destinés à un autre système de courrier électronique soient dirigés vers la passerelle. De plus, celle-ci doit être capable d'envoyer des messages.

II) la correspondance entre les champs d'en-tête et entre les modes d'adressage.

Le degré de fonctionnalité qu'une passerelle peut maintenir entre des systèmes de courrier électronique dépend de la manière dont les champs se ressemblent. Si les services fournis par les systèmes interconnectés sont similaires, alors une implémentation permettant une correspondance valable, ne donnera qu'une petite perte de fonctionnalités quand l'utilisateur communiquera avec l'autre système. Cependant, si un des deux systèmes offre un moins grand nombre de services que l'autre, alors la passerelle ne peut pas les compenser et les utilisateurs de la messagerie perfectionnée disposeront de moins de services : la passerelle ne peut pas toujours être transparente aux utilisateurs.

III) la conversion du contenu.

Si les systèmes de courrier électronique ne comprennent uniquement que du texte et utilisent des ensembles de caractères correspondants, la conversion n'est pas nécessaire. S'ils ne correspondent pas, comme dans le cas de ASCII et de EBCDIC, ou si un des systèmes permet d'autres types de contenu (graphismes, ...), le problème de conversion doit être considéré.

IV) la génération de messages d'erreurs.

Une passerelle doit être capable de générer des messages d'erreur compris par les messageries interconnectées.

CHAPITRE III : PRESENTATION DU PROBLEME.

Avec l'intérêt croissant porté aux systèmes de courrier électronique, il devenait inconcevable que différents systèmes de messagerie ne puissent communiquer parce que, non implémentés par le même concepteur, ils ne respectent pas le même protocole. C'est pourquoi, le Comité Consultatif International de Télégraphie et de Téléphonie (CCITT) a défini des normes internationales appelées les Recommandations X400 [CCITT, X400], [CCITT, X401], [CCITT, X411], [CCITT, X420].

Les concepteurs ont compris qu'ils avaient avantage à implémenter tout courrier électronique dans le respect de X400. Il est évident que la mutation des systèmes existants et respectant des syntaxes telles que RFC-822, vers un monde se référant totalement à X400 se fait de manière progressive et que l'interconnexion entre systèmes respectant des syntaxes différentes est devenue une demande prioritaire. Cette nécessité d'interconnexion est surtout d'application entre deux mondes importants de protocoles que sont, d'un côté, les normes X400 et de l'autre, le monde respectant la syntaxe RFC-822 [Crocker D., RFC-822]. La communication entre deux courriers électroniques de deux normes différentes s'effectue par l'établissement d'un système de passerelle et dans ce dessein, Steve Kille suggère une connexion entre RFC-822 et X400 dans un document appelé RFC-987 [Kille Steve, RFC-987].

Il est utile de remarquer que la notion même de message est différente suivant le monde dans lequel on se trouve : la syntaxe RFC-822 considère un message comme un ensemble d'informations uniquement sous forme de texte. Par contre, un message du monde X400 peut être composé simultanément de textes, d'images, de voix, ...

Ce chapitre a pour but de décrire de manière générale les Recommandations X400 et la syntaxe RFC-822 pour terminer par une succincte présentation de la passerelle proposée par Steve Kille.

I) Description des recommandations X400.

a) Description du modèle OSI

Les normes OSI [Tanembaum A.] sont destinées à régir l'échange d'informations dans le cadre de l'interconnexion entre systèmes de communication ouverts. Elles ont été élaborées sur base de l'utilisation de modèles abstraits : afin de spécifier les caractéristiques externes de systèmes ouverts interconnectés, chaque système ouvert réel est remplacé par un système ouvert abstrait fonctionnellement équivalent. Le comportement extérieur des systèmes ouverts abstraits est retenu pour la définition des normes des systèmes ouverts réels. Toutes les réflexions sur cette modélisation abstraite ont fait valoir la nécessité d'une technique de structuration des systèmes : **la structuration en couches**. Selon cette technique, on considère que chaque système est logiquement composé d'ensembles ordonnés de sous-systèmes représentés pour la commodité dans l'ordre vertical et constitués d'un ou de plusieurs éléments actifs appelés entités.

La réalisation d'une couche est indépendante de celle des couches inférieures et les couches adjacentes communiquent à travers leurs frontières communes. Excepté celle de rang le plus élevé, chaque couche de niveau n fournit des services aux entités de la couche n + 1 qui sont concrétisés par le biais de primitives. Le rôle de chaque couche est d'apporter une valeur ajoutée aux services fournis par la couche qui lui est inférieure. De cette manière, le niveau le plus haut se voit offrir l'ensemble des services nécessaires pour exécuter des applications à distance. Certains services sont fournis au moyen de fonctions locales à une entité et ne dépendent pas de communication entre entités paires; mais d'autres services demandent la collaboration de deux ou plusieurs entités. Il est donc indispensable de réglementer leur interaction.

La communication entre des entités de même niveau (via les services des couches inférieures) est régie par des protocoles qui sont des règles de normalisation permettant la connexion entre systèmes conçus indépendamment, mais dans le respect de ces protocoles.

L' **Organisation Internationale de Normalisation (ISO)** a modélisé l'interconnexion des systèmes ouverts, défini le modèle de référence et les normes OSI (Open System Interconnection) qui décrivent non seulement un système ouvert en sept couches mais aussi l'interconnexion entre plusieurs systèmes via un protocole pour chaque niveau (cfr figure III.1).

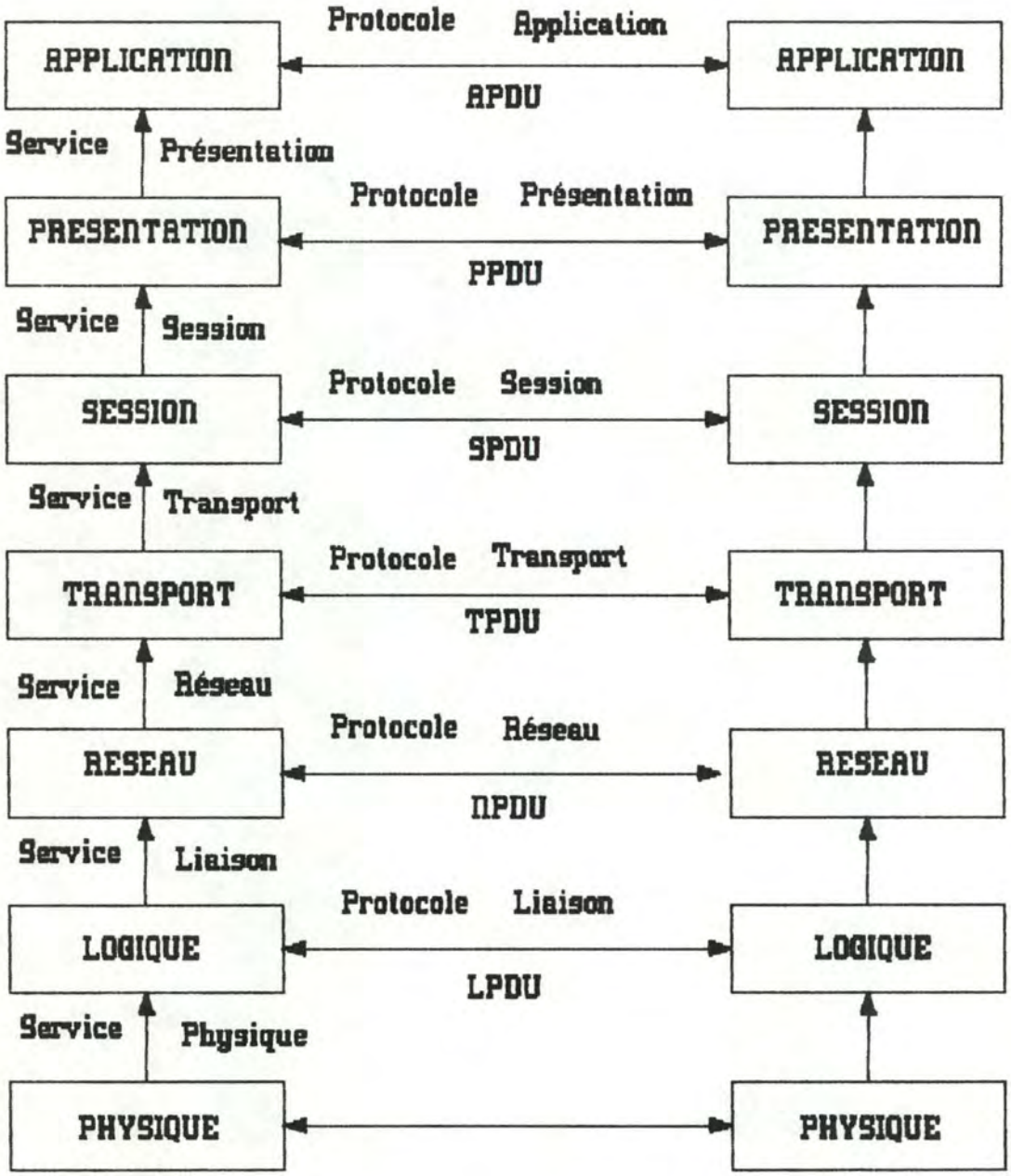


FIGURE III.1. : LES COUCHES OSI.

Une entité d'une couche de niveau n, qui désire communiquer avec une entité de même niveau dans un autre système, échange un ensemble d'informations, appelé unité de données de protocole (Protocol Data Unit : PDU). Le format du PDU est défini au bit près dans le protocole régissant la communication entre entités de la couche considérée. Le PDU est interprété comme un élément de liaison horizontale : ce lien horizontal est une description abstraite et se réalise physiquement par l'utilisation de services fournis par la couche n-1 qui envoie l'information à véhiculer à une entité de niveau n-1 de l'autre système en formant une unité de données de protocole. Cette communication horizontale au niveau n-1 s'établit elle-même par l'utilisation de services de la couche inférieure et ce jusqu'à la couche physique.

En tant que septième couche du modèle de référence d'interconnexion des systèmes ouverts, la couche APPLICATION donne le moyen d'accéder à l'environnement OSI. Son rôle est de servir de fenêtre à des programmes ou à des opérateurs humains qui désirent utiliser OSI pour communiquer ou échanger des informations. Comme exemples, nous pouvons citer les transferts de fichiers, les terminaux virtuels, les applications bancaires, la gestion de réservation de billets d'avion,... et la messagerie électronique.

b) Description en couches d'une messagerie électronique.

Le **CCITT (Comité Consultatif International de Télégraphie et de Téléphonie)** a proposé une normalisation du courrier électronique en publiant les recommandations X400 réglementant la conception de la couche APPLICATION pour la messagerie électronique. L'élaboration de ces normes s'est appuyée également sur l'utilisation de modèles abstraits et sur la technique de structuration en couches : le modèle du système de messagerie (MHS : Message Handling System) est décrit comme une division de la couche APPLICATION en deux sous-couches : **la couche UAL** (User Agent Layer) et **la couche MTL** (Message Transfer Layer).

Comme dans le modèle OSI, la fonction de la couche MTL est de fournir des services à la couche supérieure, en l'occurrence des services de transfert de messages à la couche UAL. Celle-ci offre des services à un utilisateur (programme ou personne) qui lui permettent de communiquer avec le système de transfert de messages (Message Transfer System : MTS) (cfr figure III.2).

La couche UAL se compose d'entités de même type appelées les entités d'agent utilisateur (User Agent Entity : UAE) tandis que deux types d'entités se distinguent dans la couche MTL : les entités d'agent de transfert de messages (Message Transfer Agent Entity : MTAE) et les entités de soumission et de livraison (Submission and Delivery Entity : SDE).

Une entité MTAE est un élément actif du MTS qui, en coopération avec les autres entités MTAE, a la fonctionnalité de fournir les éléments de service de transfert de messages, c'est-à-dire d'offrir l'interaction avec la couche supérieure pour l'expédition et la réception du message et de servir de relais de messages vers d'autres MTAE ou SDE.

Une entité SDE rend les services d'expédition et de réception à un UAE qui n'est pas situé sur le même système qu'un MTAE. Un SDE ne fournit pas lui-même les services de transfert de messages mais interagit avec un MTAE pour donner l'accès au MTS (cfr figure III.3).

Une entité UAE est un ensemble de processus qui, au minimum, contient les fonctions nécessaires à l'interaction avec le MTS en utilisant les services d'expédition et de réception offerts par le MTAE ou le SDE connecté. Les UAE sont regroupés en classes basées sur le type de contenu qu'ils traitent (messages interpersonnels, messages bancaires, ...). Il est indispensable que le MTS donne la possibilité à l'UAE d'identifier sa classe puisque une même classe forme une communauté séparée incapable de communiquer avec des UAE d'un autre type.

Un exemple de services offerts par la couche UAL aux utilisateurs est la messagerie interpersonnelle (Interpersonal Messaging Service) qui permet d'envoyer ou de recevoir des messages interpersonnels (IP-messages) et fournit des services d'aide à la communication avec d'autres individus ; une messagerie interpersonnelle regroupe les UAE appelés les IPM-UAE, dont les rôles sont :

- de fournir les fonctions de base nécessaires à la préparation des messages et à leur présentation aux utilisateurs,

- d'exécuter l'interaction de soumission et de livraison de messages avec le système de transfert (MTS),

- d'offrir des fonctions supplémentaires pour la facilité de l'utilisateur dans la préparation et la manipulation des IP-messages : éditeur et traitement de textes, gestionnaires de base de données enregistrant les IP-messages envoyés et reçus, ...

Nous devons faire remarquer que, comme les Recommandations X400 n'ont spécifié pour l'instant que les messages interpersonnels, nous estimons qu'il n'est plus indispensable d'alourdir la notation en ajoutant le préfixe IP- ou IPMS- devant les termes message ou UAE lorsqu'il s'agit de décrire une messagerie interpersonnelle.

Pour envoyer un message, l'utilisateur le prépare à l'aide de son UAE auquel un ordre de soumission est donné, en spécifiant le nom ou l'adresse du ou des destinataires. Le message, qui a un numéro d'identification, est alors envoyé de l'UAE expéditeur vers le(s) UAE destinataire(s) par l'utilisation des services de transfert offerts par la couche MTL.

La communication entre deux entités de même niveau dans le cadre d'un courrier électronique s'effectue suivant le principe préconisé dans le modèle OSI : les entités de la couche UAL ou MTL de deux systèmes communiquent entre elles en s'échangeant des unités de données de protocoles (PDU) appelées UAPDU ou MPDU, via les services des couches inférieures (cfr figure III.2).

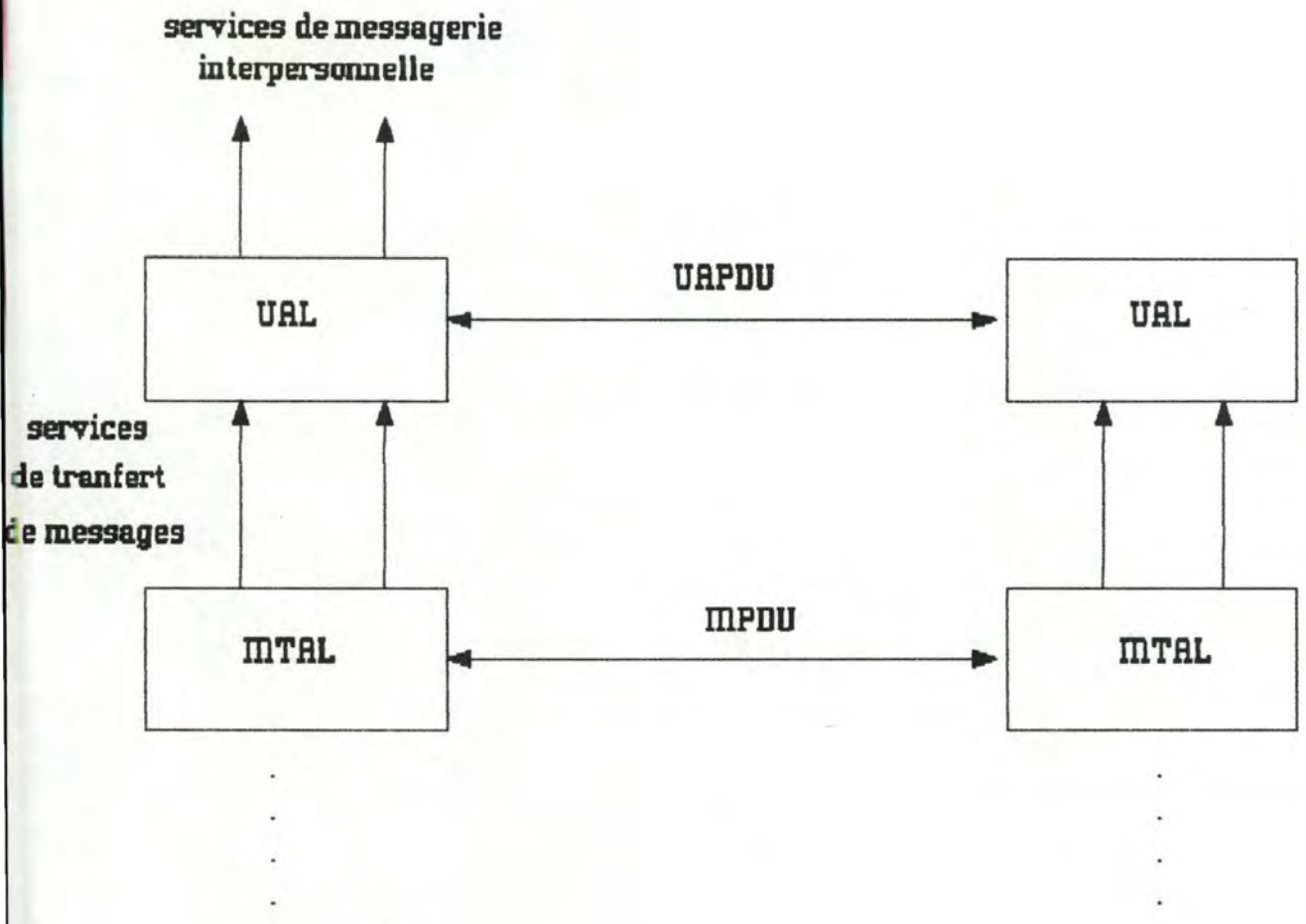


FIGURE III.2 : LA DECOUPE EN COUCHES DE L'APPLICATION DE MESSAGERIE INTERPERSONNELLE

La description d'une messagerie électronique en couches permet trois configurations :

- un système qui contient un UAE et un SDE qui interagissent avec un MTAE (S1),
- un système qui contient uniquement un MTAE (S2),
- un système qui contient un UAE et un MTAE (S3).

L'interaction entre ces trois types d'entités est régie par des protocoles décrits dans les recommandations X411 et X420 (cfr figure III.3).:

- le protocole P1 régit l'interaction entre deux entités de type MTA,
- le protocole Pc régit l'interaction entre deux entités UAE de même classe; le seul exemple existant est le protocole P2 qui régit l'interaction entre deux IPM-UAE,
- le protocole P3 régit l'interaction entre une entité de type SDE et une entité de type MTA.

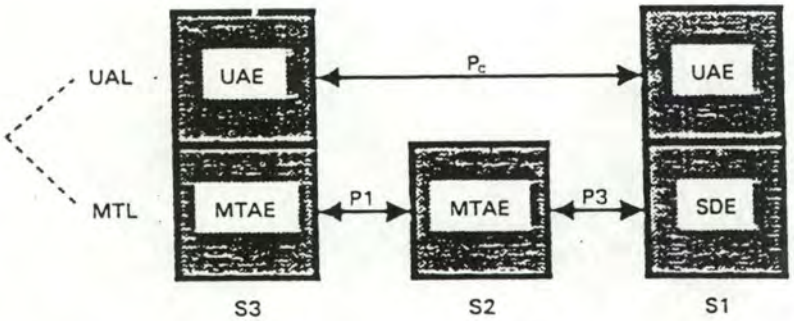


FIGURE III.3. : ENTITES DU MHS

Le protocole P1 de transfert de messages réglemente les communications entre MTAE : il définit le relais de messages entre MTAE et la manière de véhiculer demande ou réponse pour les services offerts par la couche MTL aux UAE. Quand il est transféré au moyen de P1, un message est composé d'une enveloppe et de son contenu : le contenu est l'ensemble des informations qu'un UAE expéditeur souhaite livrer à un ou plusieurs UAE destinataires; l'enveloppe contient l'information nécessaire au transfert du message ainsi que les éléments de service demandés par le UAE expéditeur.

L'interaction de relais est le moyen par lequel un MTAE transfère à un autre MTAE le contenu d'un message et l'enveloppe de relais, jusqu'à ce que le message atteigne le MTAE relié à un des UAE destinataires ou par l'intermédiaire d'un SDE.

Les éléments de protocole P1 s'appellent les Unités de Données de Protocole de message et sont de deux types :

- les User MPDU (UMPDU) transportent les messages envoyés par un UAE pour le transfert et la livraison à d'autres UAE, et les enveloppes correspondantes contenant un identificateur du message, l'expéditeur, la liste des destinataires et des informations sur la route déjà parcourue,

- les Service MPDU (SMPDU) sont utilisés pour convoyer l'information au sujet des messages entre MTAE :

PROBE MPDU : service de sonde

DELIVERY REPORT MPDU : service de rapport de livraison, ...

Le protocole P3 de soumission et de livraison permet au SDE de fournir un accès entre son UAE relié dans le système de type S1 et le MTS; les éléments de protocole P3 sont appelés les Unités de Données de Protocole des Opérations (OPDU).

Le protocole P2 est un exemple de protocole Pc réglementant l'échange d'informations entre deux entités UAE : il définit la syntaxe et la sémantique de contenu du message à expédier ou à recevoir dans le cadre d'une messagerie interpersonnelle. Les éléments de protocole P2 sont appelés les Unités de Données de Protocole d'Agent Utilisateur (UAPDU) et sont de deux types :

- les UAPDU de messagerie interpersonnelle (IM-UAPDU) qui ont deux composantes : l'en-tête qui est l'information de contrôle caractérisant le message (destinataires, priorités, ...) et le corps qui est l'information que l'utilisateur désire envoyer,

- les Status-Report UAPDU, utilisés pour indiquer la manière dont un message a été reçu.

II) Description de la norme RFC-822.

a) Définition de RFC-822.

Confronté dans son système de courrier électronique à des émissions de messages sans formalisme, le réseau américain Internet a ressenti le besoin de codifier et d'éditer, en 1977, une syntaxe unique, décrivant de manière précise un message de format texte. Le résultat de cet effort fut la norme RFC-822.

La norme RFC-822 s'applique uniquement au format de l'objet à livrer, dénommé RFC-822-message mais ne renferme aucune spécification sur son mode de livraison et de transfert. Il est supposé que l'environnement des utilisateurs comprend un système qui est capable de recevoir, de transférer et de livrer un RFC-822 message à un ou plusieurs destinataires.

b) Métasyntaxe utilisée dans RFC-822.

La syntaxe des objets manipulés dans RFC-822 est décrite selon : la forme de Backus-Naur. A cette technique de représentation s'ajoute un certain nombre de mécanismes de notations qui rendent la description des éléments plus courte et plus aisée à comprendre.

Le but de ce paragraphe est d' introduire les principales conventions d'écriture qui composent la norme RFC-822.

Pour une information plus détaillée, le lecteur se référera à [Crocker D.,RFC-822].

Les symboles généralement rencontrés sont :

- les guillemets " : ils indiquent une chaîne terminale, c'est-à-dire des caractères qui doivent apparaître tels quels dans une règle.

exemple :

La règle **mot = "\"char** précise que tout élément de type "mot" doit commencer par le caractère \.

- les symboles < et > : ils ont pour unique but de faciliter la lecture d'une règle. Leur absence ou leur présence ne change pas la définition d'un élément.

exemple :

La règle **CRLF = <CR> <LF>** est équivalente à la règle **CRLF = CR LF**.

- les parenthèses (et) : elles permettent de considérer leur contenu comme un élément autonome.

- les crochets [et] : ils renferment des éléments facultatifs.

- le symbole * : il indique le nombre d'occurrences possible d'un élément.

La forme générale est :

<l> * <m> element

où l représente le nombre minimum d'éléments et m le nombre maximum. Les valeurs par défaut sont zéro et l'infini.

exemple :

- "*(mot)" permet de zéro à une infinité d' ccurrences de "mot".

- "1*(mot)" demande au moins un élément

- "1*2"(mot) autorise un ou deux éléments.

Remarquons que "[mot]" a la même signification que "*1(mot)".

- le symbole # : il indique le nombre d'occurrences possible d'un élément, deux occurrences étant séparées par une ou plusieurs virgules.

La forme générale est :

<l> # <m> element

où l représente le nombre minimum de répétitions et m le nombre maximum. Les valeurs par défaut sont zéro et l'infini. Cette construction autorise la présence d'éléments nuls mais ceux-ci ne sont pas comptabilisés. Par exemple, "mot,,mot" est permis mais représente deux occurrences de "mot". De ce fait, "1#(mot)" exige au moins un élément non nul.

- le symbole / : il propose le choix entre différents éléments.

exemple :

La règle **mot** = x / y accepte x ou y.

c) Contenu d'un RFC-822 message.

Un RFC-822-message se compose (cfr figure III.5) :

- d'un en-tête (header) divisé en plusieurs champs (fields);

- optionnellement d'un corps (body).

L'en-tête précède le corps et la séparation entre les deux éléments est une ligne nulle c.-à-d. une ligne ne contenant rien avant le Carriage Return/Line Feed (CRLF). Le corps est simplement une séquence de lignes contenant des caractères ASCII :

RFC-822-message = *field *(CRLF *text)

text = < caractère ASCII excepté CRLF >.

CRLF = CR LF.

CR = < caractère de code ASCII décimal 13 >.

LF = < caractère de code ASCII décimal 10 >.

Cette définition présente le message comme une succession de lignes de texte. Aucune considération significative n'est donnée aux questions de compression de données ou de transmission efficace. Seuls certains champs de l'en-tête sont spécifiés dans un format rigide.

La syntaxe RFC-822 va s'attacher particulièrement à leur description. Celle-ci se fait en deux étapes : la première facilite la distinction des champs de l'en-tête entre eux, la seconde est spécifique à la structure interne d'un champ particulier. Cette séparation est destinée à permettre à un simple analyseur syntaxique d'agir sur la structure générale du message sans s'inquiéter de la structure détaillée d'un champ de l'en-tête individuel, celle-ci étant laissée à la charge d'un analyseur plus spécialisé.

Dans la première étape, chaque champ de l'en-tête est décrit comme une seule ligne logique de caractères ASCII comprenant le nom de champ (field-name) et un corps (field-body), même si, pour une plus grande lisibilité, la partie field-body peut être divisée en une représentation sur plusieurs lignes. Un champ est donc vu comme étant composé d'un nom suivi de deux-points, suivi optionnellement d'un corps et toujours terminé par un Carriage Return/Line Feed.

field = field-name ":" [field-body] CRLF.

Le nom du champ est constitué de caractères ASCII excepté les CTLs, SPACE et deux-points.

Le corps du champ est formé de caractères ASCII excluant le Carriage Return/Line Feed.

field-name = 1 * < caractère ASCII sauf CTLs, SPACE et ":" >.

field-body = *text

CTL = < caractère de code ASCII décimal de 0 à 31 >.

SPACE = < caractère ASCII de code décimal 32 >.

La seconde étape de la syntaxe RFC-822 décrit de façon plus détaillée certains champs. Leur corps, appelé field-body structuré, est alors défini différemment de "*text". Ces champs renferment des renseignements comme la date de création du message, l'adresse de(s) destinataire(s), l'adresse de l'expéditeur, etc... Dans ce dessein, une fonction importante de cette norme est la standardisation d'un format d'adresses. Le paragraphe 2 du chapitre IV est consacré à la définition d'une adresse RFC-822. Sa structure générale se présente sous la forme :

local-part "@" subdomain * ("." subdomain)

où "local-part" désigne la boîte aux lettres de l'utilisateur expéditeur ou destinataire du message et la partie à droite du caractère "@", le site.

Quelques champs au corps structuré ("date" et "from") et au corps non structuré ("subject") sont représentés à la figure III.4. Une liste détaillée est explicitée au paragraphe 2 du chapitre IV.

```
" date "    ":"   date-time

" from "    ":"   (local-part "@" domain) /
                          phrase route-addr

" subject "  ":"  *text
```

FIGURE III.4 : EXEMPLES DE CHAMPS.

Un corps structuré consiste en la combinaison d'une séquence de symboles suivants :

- special
- quoted-string
- domain-literal
- comment
- atom

field-body structuré = * < caractères ASCII formant la combinaison d'atoms, quoted-strings, domain-literals, comments, specials >.

Les " specials " sont des caractères délimitants :

specials = "(" / ")" / "<" / ">" / "@" / "," / ";" / ":" / "\" / <"> / "." / "[" / "]"

Le " quoted-string " est une suite de caractères entre des guillemets :

quoted-string =

<"> *(< caractère ASCII excepté <">, "\" et CR >
/ < caractère ASCII précédé de "\" >)
<">

Le " domain-literal " est une suite de caractères entre des crochets :

domain-literal =

"[" *(< caractère ASCII excepté <">, "\" et CR >
/ < caractère ASCII précédé de "\" >)
"]"

Le " comment " est une suite de caractères ASCII entre des parenthèses et qui n'est pas dans un quoted-string. Cette construction permet d'ajouter des informations utiles aux lecteurs :

comment =

"("*(<caractère ASCII excepté "(",")","\" et CR>
/ < caractère ASCII précédé de "\" >
/ comment)
")".

Un "atom" est un string formé de caractères ASCII excepté les " specials ", SPACE et CTLs :

atom = 1*< caractère ASCII excepté specials, SPACE,

CTLs >

Dans la définition du quoted-string, domain-literal et comment, précéder un caractère du signe "\" est un mécanisme qui permet d'utiliser des caractères généralement réservés à délimiter un symbole. Ainsi, des guillemets peuvent être insérés dans un quoted-string à condition d'être précédés d'un "\". De cette façon, ils ne seront pas considérés comme terminaison du quoted-string.

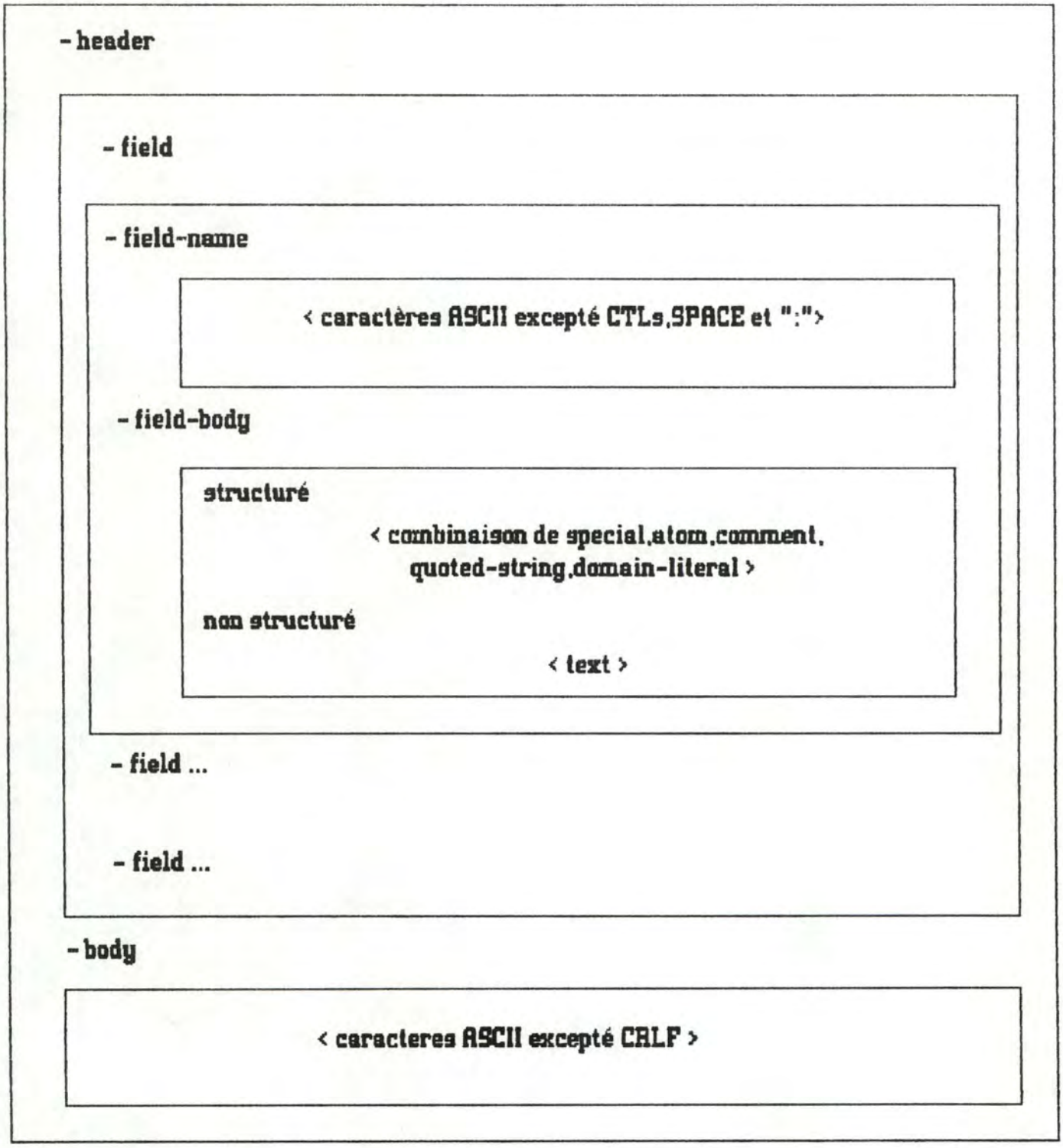


FIGURE III.5 : RFC-822 MESSAGE.

d) Environnement de RFC-822

Sur le réseau Internet, RFC-822 est couramment utilisé conjointement avec un autre standard : RFC-821, aussi connu sous le nom de Simple Mail Transfer Protocol (SMTP) [Postel J.,RFC-821]. De plus, l'adressage hiérarchique introduit syntaxiquement dans RFC-822, est défini sémantiquement dans le cadre du document RFC-920.

L'objectif du Simple Mail Transfer Protocol est de transférer le courrier de façon sûre et efficace. Son principe est construit sur le modèle de communication suivant (figure III.6) : à la demande d'un utilisateur, le sender-SMTP établit un canal de transmission vers un receiver-SMTP; celui-ci est soit un intermédiaire, soit la destination finale. Le sender-SMTP génère des SMTP commandes vers le receiver-SMTP. En réponse, le receiver-SMTP émet vers sender-SMTP des SMTP réponses. Commandes et réponses ont une syntaxe rigide.

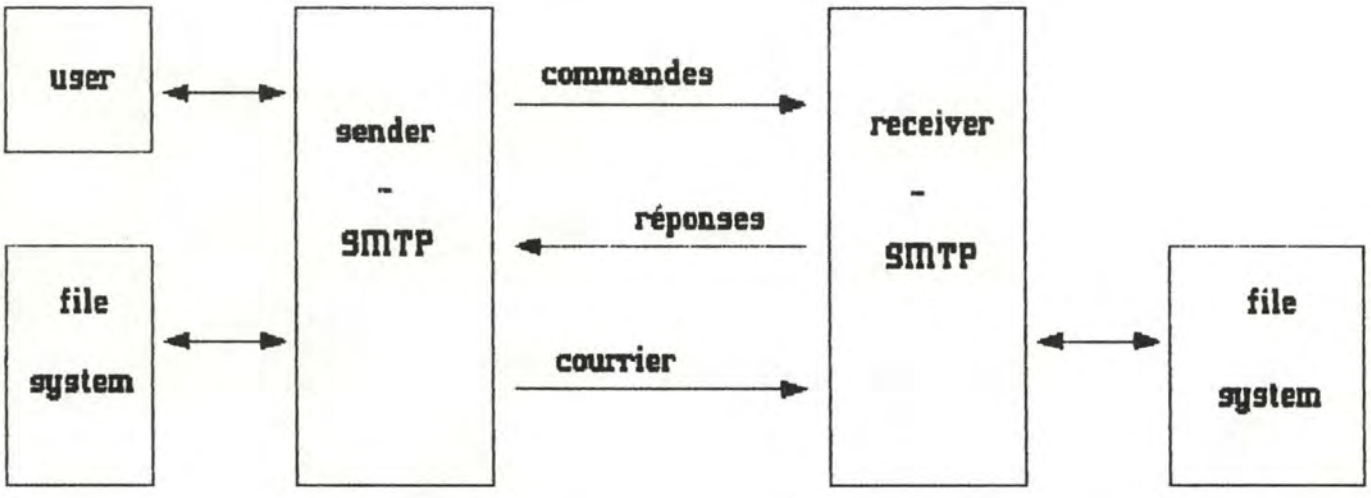


FIGURE III.6 : MODELE DU SMTP.

Lorsque le canal de transmission a été établi, le sender-SMTP émet une commande MAIL. Son argument est un reverse-path qui spécifie de qui provient le courrier. Le reverse-path est une route de retour qui peut être utilisée pour retourner le message en cas d'erreurs de transmission. En plus de l'adresse de l'expéditeur, le reverse-path peut donc contenir une liste d'hôtes. Le premier hôte, à gauche dans le reverse-path, devrait être le sendeur-SMTP envoyant la commande :

"MAIL" <space> "FROM" ":" < reverse-path > <CRLF>.

reverse-path = "<" [< liste d'hôtes > ":"] adresse">"

< liste d'hôtes > = 1#("@"domain)

adresse = local-part "@" domain

où "local-part" spécifie le nom de l'expéditeur et "domain" le site.

Exemple :

MAIL FROM : <@HostX.ARPA:Smith@HostY.ARPA>

spécifie l'émetteur de la commande (HostX relaie le message) et l'origine du couurrier (Smith situé sur HostX).

Si le receiver-SMTP accepte le MAIL, il émet une réponse favorable (OK-réponse).

Les réponses aux SMTP commandes sont conçues pour s'assurer de la synchronisation des demandes et des actions dans le processus de transfert du courrier et pour garantir que le sender-SMTP connaît toujours l'état du receiver-SMTP. A chaque commande doit correspondre exactement une réponse.

Une SMTP réponse consiste en un nombre de trois chiffres suivi d'un texte et terminé par un Carriage Return/Line Feed; le nombre contient l'information utile au sender-SMTP, le texte celle destinée à l'utilisateur.

Le premier chiffre indique la qualité de la réponse à la commande bonne, mauvaise ou incomplète; le second le domaine dans lequel la réponse se situe : syntaxe, information, connexion, système de courrier; et le dernier chiffre fournit une information plus détaillée en rapport avec le domaine spécifié.

Ainsi le code 250 indique que la commande transmise par le sender-SMTP (5) a totalement été réalisée (2). Le code 354 demande du sender-SMTP (5) des informations supplémentaires (3) dont la terminaison est déterminée par le troisième chiffre (4).

Le sender-SMTP envoie alors une commande RCPT. Son argument est un forward-path qui spécifie à qui le courrier est destiné. Le forward-path est une route source : en plus de l'adresse du destinataire, le forward-path peut contenir une liste d'hôtes. Le premier hôte, à gauche dans ce forward-path, devrait être le sender-SMTP recevant cette commande :

"RCPT" <space> "TO" ":" <forward-path> <CRLF>.

forward-path = "<" [< liste d'hôtes > ":"] adresse ">"

< liste d'hôtes > = 1#("@"domain)

adresse = local-part "@" domain

où "local-part" spécifie le nom du destinataire et "domain" le site.

EXemple :

RCPT TO : <@HostA.ARPA,@HostB.ARPA:Smith@HostD.ARPA>

spécifie le destinataire du message (Smith sur HostD) et le premier hôte relayant le message (HostA).

Le receiver-SMTP peut accepter ou refuser le courrier pour ce destinataire; suivant le cas, il émet une ok-réponse ou le rejette. Si le courrier est adressé à d'autres personnes, le même mécanisme, commande RCPT et OK-réponse, doit être repris

Quand tous les destinataires ont ainsi été négociés, le sender-SMTP signale, au moyen de la commande DATA, le début du transfert des données. Le receiver-SMTP indique alors son désaccord ou son accord. Dans ce dernier cas, toute ligne envoyée au receiver-SMTP est vue comme une partie du message. Sa terminaison est perçue par une séquence spéciale : une ligne contenant seulement un point. Le receiver-SMTP indique le succès de l'opération par une ok-réponse.

Ce modèle de communication est décrit dans l'exemple suivant :

un utilisateur nommé "Smith" et situé sur un hôte X désire communiquer un message à un utilisateur "Joe" situé en un hôte Z. Le courrier doit être relayé par un hôte Y. Un receiver-SMTP va servir de relais à la transmission du message.

pour plus de clarté, nous précédons chaque commande du nom de l'hôte qui l'émet.

X ------> Y ------> Z

X établit la connexion avec Y et entame le dialogue décrit précédemment. Il émet la commande MAIL qui spécifie de qui provient le courrier:

X : MAIL FROM : < Smith @ X.ARPA >.

Y accepte la commande et y répond favorablement selon la syntaxe adéquate:

Y : 250 OK.

X spécifie le destinataire du message par la commande RCPT et ceci de deux façons possibles : soit par l'adresse du destinataire (cas I), soit par une route (cas II) qui mène au destinataire (liste d'hôtes intermédiaires et adresse finale).

X : **RCPT TO < Joe @ Z.ARPA >** (cas 1)

ou

X : **RCPT TO : < @ Y.ARPA: Joe @ Z.ARPA >** (cas 2)

Y accepte la commande:

Y : **250 OK.**

X signale le début du transfert des données:

X : **DATA.**

Y accepte et rappelle de terminer le message avec une ligne contenant un point:

Y : **354 Start mail input; end with <CRLF>.<CRLF>.**

X transfère le message:

X : **Date ...**

From...

...

.

La fin du message est signalée, Y reprend le dialogue et accepte le message:

Y : **250 OK.**

Y doit alors transférer le courrier en Z. De receiver-SMTP, il devient sender-SMTP et établit la connexion avec Z.

Y émet la commande MAIL et modifie l'argument en insérant son identifiant au début du reverse-path. Le premier hôte à partir de la gauche dans le reverse-path est ainsi le sender-SMTP envoyant la commande.

Y : **MAIL FROM : <@ Y.ARPA: Smith @ X.ARPA>**

Z accepte la commande:

Z : **250 OK.**

Y émet la commande RCPT. Si l'identifiant de Y figurait en tête dans le forward-path de la commande RCPT émise par X (cas 2), Y l'efface du forward-path et le transmet ainsi modifié à l'hôte Z. Dans le cas contraire (cas 1), le premier hôte de la liste lui sert à déterminer le prochain receiver-SMTP destiné à recevoir le courrier. Le forward-path reste alors inchangé.

Y : **RCPT TO : < Joe @ Z.ARPA >**

Le dialogue se poursuit :

```
Z :     250 OK.

Y :     DATA

Z :     354 Start mail input; end with <CRLF>.<CRLF>

Y :     Date...

        From...

        ...

        .

Z :     250 OK.
```

RFC-920 est une spécification pour un système de noms de domaines. Il divise l'espace global américain en domaines par nature de sites : les top-domains. Les sites universitaires sont réunis au sein de EDU, les militaires au sein de MIL; les commerciaux COM etc ... et tous les sites non américains sont regroupés par pays définis par un code de deux lettres. RFC-920 définit également une procédure pour l'enregistrement d'un top-domain.

L'usage de RFC-822 ne se limite pas à la seule communauté Internet, mais l'environnement du protocole est propre à cette configuration.

Les réseaux UUCP / EUNET utilisent des extensions de RFC-822 avec des différences dans la syntaxe des adresses et n'emploient pas SMTP.

Le réseau CSNET suit le protocole RFC-822 mais utilise la norme PHONENET en remplacement de SMTP.

Le réseau BITNET/EARN utilise des formats d'adresses qui s'apparentent à RFC-822 mais avec de nombreuses adaptations.

III) Description d'une passerelle entre X400 et RFC-822 : le document RFC-987.

Le document RFC-987 [Kille Steve] présente une proposition d'interconnexion entre les systèmes exploitant les services et le protocole X400 et les systèmes de transfert de message respectant la syntaxe RFC-822. Cette interconnexion s'établit par un programme de passerelle au niveau APPLICATION, installé sur un site disposant d'un MTAE et d'un noeud du système de transfert d'un RFC-822 message.

Le but optimal d'une telle passerelle est d'offrir tous les services de la norme du monde dans lequel se trouve un utilisateur, indépendamment du passage de son message dans un autre monde. des messages peuvent également être transférés plusieurs fois d'un monde à l'autre et il faut donc que les correspondances soient réversibles.

La différence du mode d'adressage et la diversité des services offerts par les normes X400 et la syntaxe RFC-822 rendent très complexe la réalisation d'une interconnexion. La présentation même des syntaxes est différente et augmente la difficulté de recherche de correspondances sémantiques. Les Recommandations X400 définissent les éléments de service offerts aux utilisateurs et spécifient les éléments de protocole qui permettent leur implémentation. La syntaxe RFC-822 présente uniquement un message comme un contenu précédé d'un en-tête dont les champs se rapprochent des éléments de protocole X400.

Pour spécifier les fonctions de la passerelle, nous devons établir la correspondance sémantique entre les éléments de protocole au sens X400 et les champs d'un RFC-822. Pour les éléments ou les champs qui n'ont pas de correspondants, certains artifices seront utilisés tels que la création de champs d'extension ou l'insertion dans le corps du message.

a)Hypothèses sur X400.

Deux couches de services sont décrites dans X400 :

- le MESSAGE TRANSFER SERVICE (MTS) qui peut être fourni par les protocoles P1 ou P3 spécifiés dans X411

- l' INTERPERSONAL MESSAGING SERVICE (IPMS) qui est fourni par le protocole P2 spécifié dans X420.

Rappelons que ces deux protocoles consistent essentiellement en la définition d'un ensemble d'éléments de protocoles ayant chacun une syntaxe et une sémantique standardisées qui peuvent être utilisées pour construire, pour P1, les MPDU et pour P2, les UAPDU.

Trois types de MPDU sont définis :

- les User MPDU,

- les PROBE MPDU,

- les Delivery Report MPDU.

Deux types de UAPDU sont définis :

- LES IM-UAPDU,

- les SR-UAPDU.

b)Hypothèses sur RFC-822.

Chaque réseau comme ARPANET, UUCP, BITNET utilise RFC-822 avec un support de transfert de courrier spécifique à sa configuration. RFC-987 fait l'hypothèse que le protocole RFC-822 est basé sur un service appelé le service 822-P1 qui lui fournit les trois fonctions de base suivantes :

1 : l'identification d'une liste de destinataires,

2 : l'identification d'une adresse de retour en cas d'erreurs,

3 : le transfert du RFC-822-message,

auxquelles correspondent les éléments de protocoles suivants :

- le 822-P1 Originator address,

- le 822-P1 recipient.

Un exemple de service 822-P1 est SMTP utilisé dans le réseau Internet. Les informations 822-P1 Originator address et 822-P1 recipient dénotent respectivement le forward-path et le return-path. SMTP peut fournir des fonctions supplémentaires. Celles-ci ne sont pas communes à tout protocole 822-P1, nous ne les envisageons donc pas ici.

c)Nature de la passerelle.

La nature fonctionnelle de la passerelle peut maintenant être précisée. Une première idée (figure III.7) est de faire correspondre aux services 822-P1 le protocole P1 et à RFC-822 le protocole P2. Cette approche ne s'adapte pas à la réalité car aux éléments de P1 et de P2 correspondent des éléments de RFC-822 et 822-P1 et inversément.

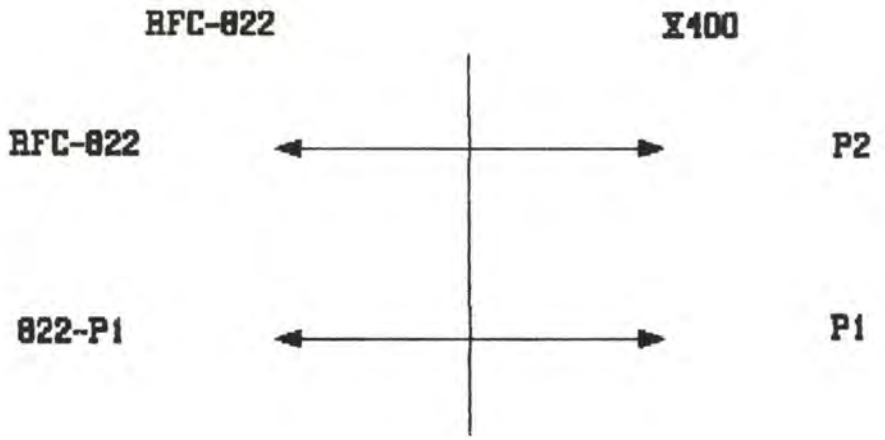


FIGURE III.7 : PROPOSITION REJETEE DE PASSERELLE.

Steve Kille propose la correspondance suivante (figure III.8) : de RFC-822 vers X400, le RFC-822 message et l'information 822-P1 associée sont toujours mis dans un IM-UAPDU avec l'enveloppe P1 associée. De X400 vers RFC-822, le RFC-822 message et l'information 822-P1 associée sont dérivés de

1. un DELIVERY REPORT MPDU,
2. un SR-UAPDU et l'enveloppe P1 associée,
3. un IM-UAPDU et l'enveloppe P1 associée.

Les PROBE MPDU doivent être traités par la passerelle tandis que tout autre MPDU est rejeté à la passerelle.

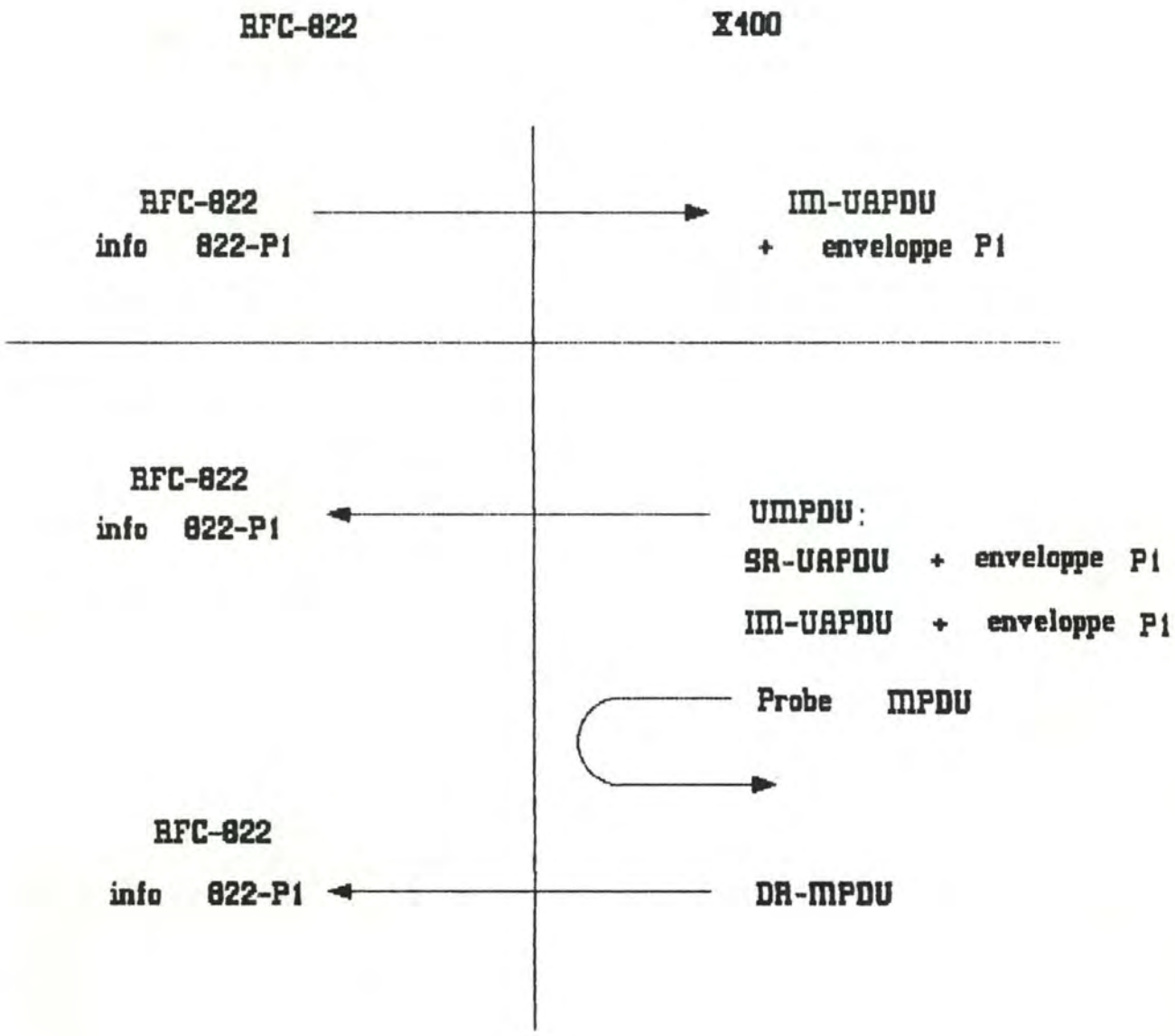


FIGURE III.8 : CORRESPONDANCES ENTRE RFC-822 ET X400.

Le chapitre IV présentera la correspondance des adresses et le chapitre V décrira de façon plus explicite les correspondances au niveau des différents éléments de services.

CHAPITRE IV : CORRESPONDANCE DES ADRESSES.

Nous abordons dans ce chapitre le problème le plus épineux posé par l'élaboration d'une passerelle RFC-987 : la correspondance des adresses.

Chacun des deux mondes de protocoles proposent une syntaxe et des formats d'adresses très différents. La norme X400 présente les attributs d'une adresse X400 et leur composition; la norme X409 une technique de spécification et d'encodage. De son côté, le standard RFC-822 définit une adresse à l'aide du métalangage BNF (Backus-Naur Form).

Le début de ce chapitre reprend les éléments principaux de ces recommandations en fonction desquels est élaborée la correspondance des adresses. Cette dernière est alors décrite sur base des documents RFC-987 [Kille S.] et de Ruediger Grimm [Grimm R].

I) L'adressage selon les Recommandations X400.

a) Spécification d'un ORName [CCITT,X400].

Un utilisateur est spécifié par un nom global et unique: le nom descriptif (descriptive name).

Pour rester dans le contexte du MHS et comme chaque utilisateur est servi par un UA, les entités désignées sont les UAE. Dès lors, le nom descriptif de tout utilisateur est associé à celui de l'UA et est appelé Originator Recipient Name (ORName).

Un UAE appartient a un Management Domain (MD). Un MD est une organisation qui contrôle une portion du MHS. Celle-ci peut être privée (PRivate Management Domain) ou publique (ADministration Management Domain). Chaque MD doit s'assurer que chaque UAE dans le MD a un nom unique appelé nom primitif (primitive name).

Un ORName va se composer d'un ensemble minimum d'attributs de base (base attribute set) dont les valeurs identifient sans ambiguité le MD particulier qui sert l'utilisateur et d'un ensemble d'attributs constituant le nom primitif.

La structure d'un ORName est donc une liste d'attributs et de valeurs associées. De cette façon, une sélection soigneuse dans cette liste permet d'identifier de manière unique un simple UAE avec un petit nombre de paires attribut/valeur.

Un UAE est également spécifié par son adresse appelée OR address. Une OR address est un ORName qui contient certaines caractéristiques qui aident le MTS à situer l'UAE dans le MHS global. Elle spécifie le point d'attachement de l'UAE plutôt que son nom.

Actuellement, les recommandations X400 [CCITT,X400] définissent une liste d'attributs standards desquels sont tirées deux formes de ORName qui sont en fait des OR addresses.

La première forme spécifie en deux phases le destinataire ou l'expéditeur du message : d'une part, par le nom du pays dans lequel le domaine public est situé et par le nom de ce domaine dans ce pays (base attribute set), et d'autre part, par un ensemble d'attributs et de valeurs suffisant pour identifier l'utilisateur dans le domaine spécifié (primitive name).

Parmi cet ensemble, trois variantes sont permises :

- la première variante consiste en une collection de un ou plusieurs attributs choisis parmi le nom personnel, le nom d'une organisation, les noms d'unités organisationnelles, et les noms de domaines privés.

- la deuxième variante consiste en un identificateur qui distingue l'UA de l'utilisateur de tous les autres UAs servis par le même MD.

- la troisième variante consiste en une adresse X121 respectant les Recommandations X121.

De ces trois variantes sont tirées les trois ORNames :

```
ORName =    CountryName
            AdministrationDomainName
            [PrivateDomainName]
            [PersonalName]
            [OrganizationName]
            [OrganizationalUnit]
            [DomainDefinedAttribute]

ORName = CountryName
         AdministrationDomainName
         UniqueUAIdentifier
         [DomainDefinedAttribute]

ORName = CountryName
         AdministrationDomainName
         X121Address
         [DomainDefinedAttribute]
```

Les normes X400 tolèrent les attributs définis par une administration : les Domain Defined Attributes. Ces attributs représentent des structures de noms développées par les systèmes existant avant l'élaboration des recommandations X400. L'inclusion des Domain Defined Attributes reflète le désir de créer un MHS reliant les MDs existants. Il est attendu que ces systèmes évoluent vers l'usage des attributs standards.

La seconde forme d'un O/R name spécifie l'expéditeur ou le destinataire d'un message par son adresse télématique. Elle consiste en, soit une adresse X121, soit une adresse X121 suivie d'un identificateur de terminal.

ORName = X121Address
[TerminalID]

b) Syntaxe d'un ORName [CCITT,X409].

Chaque attribut d'un OR-Name possède un type et une valeur. Un type de donnée est une classe de l'information (numérique, texte). Une valeur de donnée est un représentant d'une telle classe (un nombre, un fragment de texte).

A chaque type correspondent une notation standard et une représentation standard. La notation standard est l'ensemble des conventions utilisées pour décrire soit le type lui-même, soit une valeur de ce type. La représentation standard est l'ensemble des règles pour encoder en une séquence d'octets, les objets décrits sous notation standard.

b.I. La représentation standard.

La représentation standard d'une valeur d'un type est un élément de donnée. Un élément de donnée est une séquence entière de n octets numérotés de 0 à n-1, chaque octet constituant une séquence de huit bits numérotés de 8 à 1. Il est formé de trois composantes dans l'ordre suivant : la composante "Identifier", la composante "Length" et la composante "Contents".

b.I.1. "Identifier".

La composante "Identifier" différencie un type d'un autre et gouverne l'interprétation de la composante "Contents".

L'octet numéro 0 permet de distinguer :

- au moyen des bits 8 et 7, la classe qui définit l'application particulière du type.

  La classe "Universal" est encodée en 00, la classe "Application-wide" en 01, la classe "Context-specific" en 10 et la classe "Private-use" en 11.

- au moyen du bit 6, la forme de l'élément de données.

  La forme "primitive" est encodée en 0, la forme "constructor" en 1. Un élément de forme "primitive" est celui dont la partie "Contents" est atomique, un élément de forme "Constructor" est celui dont la partie "Contents" est elle-même un élément de donnée.

- au moyen des bits 5 à 1, le ID-code qui distingue un type de donnée d'un autre de la même classe.

b.I.2. "Length".

La composante "Length" spécifie la longueur en octets de la composante "Contents". Elle peut prendre trois formes : courte, longue et infinie. Par simplicité, la forme courte sera préférée : elle est constituée d'un octet dont le bit 8 vaut 0 et dont les bits 7 à 1 déterminent le nombre d'octets, inférieur à 128, de la composante "Contents".

b.I.3. "Contents".

La composante "Contents" contient l'information que l'élément est destiné à véhiculer.

A titre d'exemple, nous donnons la représentation standard d'une valeur des types NumericString et PrintableString .

Le type NumericString représente un ensemble ordonné de zéro ou plusieurs caractères qui collectivement encodent une information sous forme textuelle. Sa classe est "universal" et son ID-code vaut 18.

La représentation standard de la valeur "23" de type NumericString vaut :

Identifier	length	Contents	
00010010	00000010	00110010	00110011

car la classe est "universal" (deux premiers bits du premier octet = 00), la forme est "primitive" (troisième bit du premier octet = 0), l'ID-code vaut 18 (les cinq derniers bits du premier octet = 10010), le nombre d'octets du contenu est 2 (deuxième octet) et le contenu (troisième et quatrième octets) est "2" (code ASCII 50) et "3" (code ASCII 51).

Le type PrintableString représente un ensemble ordonné de zéro ou plusieurs caractères choisis dans un sous-ensemble de caractères imprimables. Sa classe est "universal" et son ID-code vaut 19.

La représentation standard de la valeur "T" (code ASCII 84) de type PrintableString vaut :

Identifier	Length	Contents
00010011	00000001	01010100

b.II. La notation standard.

La notation standard fournit un ensemble de règles syntaxiques permettant de définir des types de données. Elle fournit également un certain nombre de types de base, tels que integer, boolean et des constructeurs de types, tels que Sequence et Tagged. Les règles syntaxiques procurent des mécanismes permettant la définition de nouveaux types à partir des types de base et des constructeurs de type, et la description de valeurs de ces types. Ainsi, la recommandation X409 définit certains types utilisables universellement, tels que IA5String, Numericstring, etc.

Seuls les types intervenant dans la composition d'un ORName (figure IV.1.) sont décrits dans ce paragraphe. Le lecteur se réfèrera à [CCITT,X409] pour de plus amples informations.

b.II.1. Constructeur de types.

b.II.1.1. Le type SEQUENCE.

Une séquence représente un ensemble ordonné de zéro ou plusieurs valeurs appelées éléments de la séquence. Elle peut se composer d'un nombre variable d'éléments de même type ou d'un nombre fixe d'éléments d'un ou plusieurs types.

Une séquence de multiples types peut avoir des éléments facultatifs. Ces éléments sont identifiés au moyen du mot clef OPTIONAL.

SequenceType ::= SEQUENCE / SEQUENCE OF TYPE / SEQUENCE{ElementsTypes}

Sa représentation est un élément dont la classe est "Universal", la forme "constructor", l'ID-Code 16 et dont le contenu est formé par les éléments ordonnés de la séquence dans leur représentation standard.

Exemple :

```
Nom ::= SEQUENCE{
        prenom IA5String,
        initiale IA5String,
        nomdefamille IA5String}
```

b.II.1.2. Le type SET.

Un "Set" représente un ensemble non ordonné de zéro ou plusieurs valeurs appelées membres du set. Il peut se composer d'un nombre variable de membres de même type ou d'un nombre fixé de membres d'un ou plusieurs types.

Un set de multiples types peut avoir des membres facultatifs. Ces membres sont identifiés au moyen du mot clef OPTIONAL.

SetType ::= SET / SET OF Type / SET{MemberTypes}

Sa représentation est un élément dont la classe est "Universal", la forme "constructor",l'ID-Code 17 et dont le contenu est formé par les membres du SET, dans n'importe quel ordre mais dans leur représentation standard.

Exemple :

```
Date ::= SET{
         jour [0] IA5String,
         mois [1] IA5String,
         annee [2] INTEGER OPTIONAL}
```

b.II.1.3. Le type CHOICE.

Un "CHOICE" représente une valeur dont le type est choisi parmi une ou plusieurs possibilités distinctes.

ChoiceType ::= CHOICE{AlternativeTypeList}

Exemple :

```
Sexe ::= CHOICE{
         masculin [0] IA5String,
         feminin [1] IA5String}
```

b.II.1.4. Le type "Tagged".

Le type "Tagged" fournit le moyen de créer d'autres types à partir de types existants.

TaggedType ::= Tag IMPLICIT Type / Tag Type

Tag ::= [Class number]

Class ::= UNIVERSAL / APPLICATION / PRIVATE / empty

"Class" spécifie la classe du type créé; elle est context-free par défaut, Number spécifie l'ID-Code. Le type lié au type créé peut être explicite ou implicite selon que sa représentation ou seulement son contenu est dans le contenu du nouveau type. Dans ce dernier cas le mot clef IMPLICIT apparaît dans la spécification.

Ainsi le type prenom défini comme

prenom = [APPLICATION 3] IA5String

à partir du type IA5String qui est de classe Universal et de ID-Code 22, a comme représentation standard de la valeur "Jones" :

nom	Length	Contents		
01100011	00000111	IA5String	Length	Contents
		00010110	00000101	JONES

où, ici, le contenu "Jones" n'est pas exprimé en octets;

et le type prenom défini comme

prenom = [APPLICATION 3] IMPLICIT IA5String

a sa valeur "Jones" en représentation standard :

nom	Length	Contents
01000011	00000101	JONES

b.II.2. Types définis.

b.II.2.1. Le type NumericString.

NumericString ::= [UNIVERSAL 18] IMPLICIT IA5String

Les caractères permis sont : 0,1,2,3,4,5,6,7,8,9.

II.2.2.Le type PrintableString.

Printablestring ::= [UNIVERSAL 19]IMPLICIT IA5String

Les caractères permis sont : A,B,...Z
a,b,...z
0,1,...9
space
' ( ) + , - . / : = ?

c) Exemple de représentation standard et de notation standard d'un ORName.

```
ORName                          ::= [APPLICATION 0] IMPLICIT SEQUENCE {
        StandardAttributeList,
        DomainDefinedAttributeList OPTIONAL}
StandardAttributeList           ::= SEQUENCE {
        CountryName OPTIONAL,
        AdministrationDomainName OPTIONAL,
        [0] IMPLICIT X121Address OPTIONAL,
        [1] IMPLICIT TerminalID OPTIONAL,
        [2] PrivateDomainName OPTIONAL,
        [3] IMPLICIT OrganizationName OPTIONAL,
        [4] IMPLICIT UniqueUAIdentifier OPTIONAL,
        [5] IMPLICIT PersonalName OPTIONAL,
        [6] IMPLICIT SEQUENCE OF OrganizationalUnit OPTIONAL}
DomainDefinedAttributeList      ::= SEQUENCE OF DomainDefinedAttribute
DomainDefinedAttribute          ::= SEQUENCE {
        type PrintableString,
        value PrintableString}
CountryName                     ::= [APPLICATION 1] CHOICE {
        NumericString,
        PrintableString}
AdministrationDomainName        ::= [APPLICATION 2] CHOICE {
        NumericString,
        PrintableString}
X121Address                     ::= NumericString
TerminalID                      ::= PrintableString
OrganizationName                ::= PrintableString
UniqueUAIdentifier              ::= NumericString
PersonalName                    ::= SET {
        surName [0] IMPLICIT PrintableString,
        givenName [1] IMPLICIT PrintableString OPTIONAL,
        initials [2] IMPLICIT PrintableString OPTIONAL,
        generationQualifier [3] IMPLICIT PrintableString OPTIONAL}
OrganizationalUnit              ::= PrintableString
PrivateDomainName               ::= CHOICE {
        NumericString,
        PrintableString}
```

FIGURE IV.1. : NOTATION STANDARD D'UN ORNAME.

la notation standard de l'ORName dont CountryName vaut "US", AdministrationDomainName "ATT", OrganizationName "xerox", OrganizationalUnit "Marketing", surName "Linnimouth", initials "J" et generationQualifier "S" a la forme suivante :

```
{
{CountryName"US",
AdministrationDomainName"ATT",
OrganizationName"xerox",
{surname "Linnimouth
Initials "J",
GenerationQualifier "S"},
{OrganizationalUnit"Marketing"}}}
```

Sa représentation standard vaut :

```
ORName
01100000
  Length      Contents
  00110111    StandardAttributeList
    (55)      00110000
                Length      Contents
                00110101    CountryName
                  (53)      01100001
                              Length      Contents
                              00000100    PrintableString
                                (4)       00010011
                                            Length      Contents
                                            00000010    US
                                              (2)
                            AdministrationDomainName
                            01100010
                              Length      Contents
                              00000101    PrintableString
                                (5)       00010011
                                            Length      Contents
                                            00000011    ATT
                                              (3)
                            [3]
                            10000011
                              Length      Contents
                              00000101    Xerox
                                (5)
                            [5]
                            10100101
                              Length      Contents
                              00100010    Surname
                                (18)      10000000
                                            Length      Contents
                                            00001010    Linnimouth
                                              (10)
                                          initials
                                          10000010
                                            Length      Contents
                                            00000001    J
                                              (1)
                                          generationQualifier
                                          10000011
                                            Length      Contents
                                            00000001    5
                                              (1)
                            [6]
                            10100110
                            Length      Contents
                            00001011    OrganizationalUnit
                              (11)      00010011
                                          Length      Contents
                                          00001001    Marketing
                                            (9)
```

II) L'adressage selon la norme RFC-822 [Crocker D.].

a) Adresse spécifique.

Une adresse RFC-822 consiste en une partie locale et une partie domaine séparées par un symbole "@", cette dernière étant formée de un ou plusieurs domaines séparés par un point. Au moins un domaine doit toujours être présent.

addr-spec = local-part "@" domain

local-part = word *("."word)

domain = sub-domain *("."sub-domain)

sub-domain = domain-ref / domain-literal

domain-ref = atom

"local-part" représente le nom de l'utilisateur et chaque "sub-domain" le nom d'entités administratives. Le but des domaines est d'établir une hiérarchie dans l'espace d'adressage. Un domaine est caractérisé par une "autorité de nomination" (naming authority) dont la fonction est d'attribuer les noms au sein de l'espace dont elle a la charge, tout en garantissant l'unicité [Postel Reynolds, RFC-920].

On établit ainsi une arborescence dont chaque arc définit un lien hiérarchique. Tout chemin partant de la racine à une feuille quelconque de l'arbre, définit une et une seule adresse constituée des noeuds rencontrés. Une adresse RFC-822 se comprend donc comme la localisation d'une boîte-aux-lettres dans une structure administrative, sans rapport avec sa localisation géographique.

Une adresse se lit de la droite vers la gauche. L'élément le plus à droite dénote un fils de la racine: un "top-domain". La racine n'est pas référencée car elle est commune à toute adresse. L'élément à gauche du top-domain définit un de ses sous-domaines et ainsi de suite en descendant jusqu'au dernier domaine figurant le plus à gauche dans la partie domaine où l'utilisateur "local-part" doit se trouver.

Ainsi J.Linnimouth@Marketing.Xerox.COM adresse l'utilisateur J.Linnimouth au sein du domaine Marketing, sous-domaine de Xerox, sous-domaine de COM.

Une adresse RFC-822 ne spécifie donc pas une route pour atteindre un utilisateur donné mais décrit une localisation dans l'espace d'adressage.

La gestion d'un domaine implique le contrôle de l'unicité des noms au sein de ce domaine et la fourniture des informations nécessaires pour accéder aux utilisateurs à la fois à l'intérieur et à l'extérieur du domaine.

Pour Internet, ces procédures sont décrites dans le document RFC-920. Une réglementation stricte pour l'enregistrement des top-domains y est également définie.

Rappelons que les principaux top-domains américains sont edu, mil, com et gov; et les sites non américains sont regroupés par pays définis par un code de deux lettres, le top-domain défini pour la Belgique étant dénoté "be".

Ces top-domains sont enregistrés au Network Information Center (NIC). Seule une organisation satisfaisant les critères spécifiés dans le document RFC-920 peut s'y faire enregistrer

b) Spécification d'une route.

Parfois un utilisateur peut désirer indiquer la route qu'un message devrait suivre. Dans ce dessein, RFC-822 définit la syntaxe suivante :

route-addr = "<"[route] addr-spec ">"

route = 1#("@"domain) ":"

La portion "[route]" contient cette information. Elle spécifie une séquence d'hôtes à traverser.

Cette syntaxe est également utilisée par SMTP pour représenter dans le reverse-path la route empruntée par le message et dans le return-path la route qu'il doit suivre.

c) Spécification d'une boîte-aux-lettres.

Une spécification de boîte-aux-lettres répond à la syntaxe suivante :

mailbox = addr-spec / phrase route-addr

phrase = 1*word

"phrase" est habituellement utilisé pour indiquer de façon compréhensible le nom du destinataire. Il n'est pas interprété par le système de courrier.

Ainsi les adresses :
Alfred Neuman<Neuman@BBN-TENEXA> et Neuman@BBN-TENEXA
ont toutes les deux la même sémantique.

d) Spécification multiple.

RFC-822 permet l'envoi d'un message à un ensemble de personnes. La spécification d'une tel mécanisme est facilitée par la condition :

group = phrase ":" [#mailbox] ";"

Les boîtes-aux-lettres des destinataires sont définies dans la partie encadrée par ":" et ";".Une copie du message est expédiée à chaque membre énuméré.

e)Spécification d'une adresse.

La forme la plus générale d'une adresse spécifiée dans un champ est :

address = mailbox / group

III) Correspondance entre adresses X400 et adresses RFC-822.

La correspondance entre adresses X400 et adresses RFC-822 est sans nul doute la tâche la plus complexe d'un système de passerelle. Son approche est structurée, ci-dessous, en deux parties.

La première aborde l'ensemble des problèmes liés à la syntaxe des différents protocoles et la seconde décrit la correspondance propre aux adresses des deux mondes

a) Problèmes de syntaxe.

1).Définitions.

1.1. X400.

Pour mieux comprendre la correspondance entre adresses RFC-822 et X400, les éléments d'un ORName sont redéfinis dans RFC-987 [Kille S.] à l'aide de la notation utilisée dans RFC-822.

Les types "définis" Printablestring et Numericstring sont spécifiés de la façon suivante :

```
numericstring   = *DIGIT

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

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

ps-delim        = "("
```

De même, les types et valeurs des éléments d'un ORName (figure IV.1.) sont encodés comme :

```
standard-type        =
        "C"                  ;CountryName
      / "A"                  ;AdministrationDomainName
      / "P"                  ;PrivateDomainName
      / "X121"               ;X121Address
      / "T-ID"               ;TerminalID
      / "O"                  ;OrganizationName
      / "OU"                 ;OrganizationalUnit
      / "UA-ID"              ;UniqueUAIdentifier
      / "S"                  ;PersonalName.surName
      / "GI"                 ;PersonalName.givenName
      / "I"                  ;PersonalName.initials
      / "GQ"                 ;PersonalName.generationQualifier
dd.type              =
     printablestring         ;DomainDefinedAttribute.type
standard-dd-type =
        "RFC-822"            ;dd.type = "RFC-822"
      / "JNT-Mail"           ;dd.type = "JNT-Mail"
      / "UUCP"               ;dd.type = "UUCP"

attributevalue       = c / a / p / x121 / t-id / o / ou /
                       ua-id / pn.i / pn.g / pn.s / pn.gq /
                       dd.value
c      = printablestring    ;Countryname
a      = printablestring    ;AdministrationDomainName
p      = printablestring    ;PrivateDomainName
x121   = numericstring      ;X121Address
t-id   = numericstring      ;TerminalID
o      = printablestring    ;OrganizationName
ou     = printablestring    ;OrganizationalUnit
ua-id  = numericstring      ;UniqueUAIdentifier
pn.s   = printablestring    ;PersonalName.surName
pn.g   = printablestring    ;PersonalName.givenName
pn.c   = printablestring    ;PersonalName.initials
pn.gq  = printablestring    ;PersonalName.generationQualifier
dd.value =printablestring ;DomainDefinedAttribute.value
```

1.2. RFC-822.

Une adresse de RFC-822 est définie dans RFC-987 comme:

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

addr-spec       = localpart"@"domain

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

domain-syntax   = ALPHA [*alphanumhyphen alphanum]

alphanum        = <ALPHA ou DIGIT >

alphanumhyphen = <ALPHA ou DIGIT ou "-">
```

Bien que la norme RFC-822 autorise une syntaxe plus générale pour les domaines, ce choix suit la définition la plus souvent adoptée par les services d'enregistrement de domaines.

2) Correspondance ASCII - Printablestring.

La définition d'un PrintableString ne regroupe qu'une partie de l'ensemble des caractères ASCII (figure IV.2.). Une correspondance entre une addr-spec et un ORName doit donc veiller à ce que tout caractère ASCII soit transformé en PrintableString. Cette traduction est déterminée par la syntaxe ps-encoded

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

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

le terme "3DIGIT" correspond au code décimal du caractère ASCII à traduire.

Pour encoder un string ASCII en Printablestring, tout caractère qui n'est pas analysé comme ps-delim ou ps-char est transformé selon la syntaxe ps-encoded, sinon il reste inchangé.

Ainsi

ASCII	est encodé	Printablestring
@	----->	(a)
(	----->	(
(a)	----->	(a)
+	----->	+
(040)	----->	(040)

Pour encoder un Printablestring en ASCII, tout caractère qui est analysé comme ps-encoded-char subit la transformation inverse sinon il reste inchangé.

Ainsi

Printablestring	est encodé	ASCII
(a)	----->	@
(	----->	(
(040)	----->	(
+	----->	+
((a)	----->	((a)

Cette correspondance n'est pas symétrique. En effet

ASCII		Printablestring		ASCII
(a)	----->	(a)	----->	@
(b)	----->	(b)	----->	!
(040)	----->	(040)	----->	(

3) Correspondance local-part - ASCII.

Certains réseaux basés sur RFC-822 ne peuvent traiter le format quoted-string qui apparaît dans la partie locale d'une adresse RFC-822 et qui se restreint à un sous-ensemble de caractères ASCII (figure IV.2.).

Printablestring	local-part
chiffre	CTL
lettre	<
'	>
+	@
-	[
/	]
=	;
?	"
	\
SPACE	SPACE
:	:
.	.
,	,
)	)
(	(

FIGURE IV.2. : CARACTERES PRESENTS DANS UN PRINTABLESTRING ET CARACTERES ABSENTS DANS UNE PARTIE LOCALE.

La spécification d'une local-part est alors :

local-part = atom *("."atom)

atom = 1*<caractères ASCII excepté "(" / ")" / "<" / ">" / "@" / "," / ";" / ":" / "\" / <"> / "." / "[" / "]" , SPACE et CTLs>

Pour permettre les échanges entre RFC-822 et ses variantes qui ne supportent pas le quoted-string, un mécanisme est décrit au moyen de la syntaxe atom-encoded.

```
atom-encoded = *(a-char/a-encoded-char)

a-char       = <caractère excepté specials, SPACE, CTLs, "-
               ", et "#">

a-encoded-char =
        "-"                          ; (space)
      / "#u#"                        ; (-)
      / "#l#"                        ; <(>
      / "#r#"                        ; <)>
      / "#m#"                        ; (,)
      / "#c#"                        ; (:)
      / "#b#"                        ; (\)
      / "#h#"                        ; (#)
      / "#e#"                        ; ($=)
      / "#s#"                        ; ($/)
      / "#" 3DIGIT "#"
```

Le terme "3DIGIT" correspond au code décimal du caractère ASCII à traduire.

Pour encoder un ASCII en local-part, tout caractère qui n'est pas analysé comme un a-char à l'exception du point est transformé selon la syntaxe atom-encoded ; sinon il reste inchangé.

Ainsi

ASCII	est encodé	local-part
#	----->	#h#
(	----->	#l#
a	----->	a

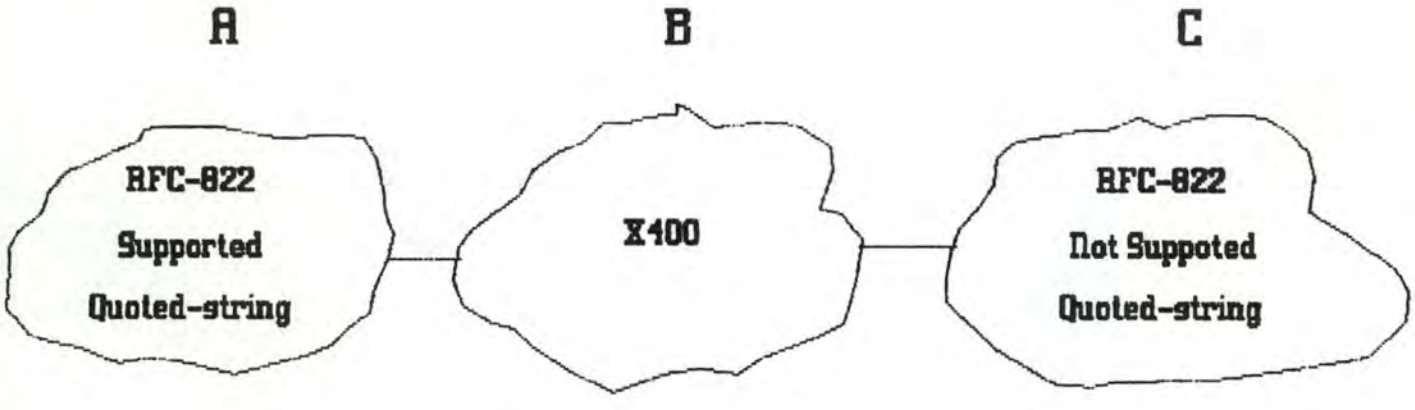
Pour encoder un printablestring en local-part, si la local-part est analysée comme atom *("."atom) et chaque atom comme atom-encoded, alors chaque caractère à l'exception du point subit la transformation inverse. Dans les autres cas, les caractères demeurent inchangés.

Ainsi

local-part	est encodé	ASCII
.	----->	.
#	----->	#
#h#	----->	#
#l#	----->	(

Au niveau d'une passerelle X400 vers RFC-822, ce mécanisme est appliqué sur toute adresse si le réseau RFC-822 cible ne supporte pas le quoted-string. Dans le sens RFC-822 vers X400, la transformation est effectuée sur toute adresse lorsque le réseau source ne supporte pas le quoted-string mais également pour toute adresse 822-P1-recipient, quel que soit le réseau source.

En effet, supposons la configuration suivante :



Un utilisateur de A désire envoyer un message vers C. Comme nous le verrons dans la suite de ce chapitre, l'adresse 822-P1 Recipient peut avoir la forme :

/C=UK/A=BT/P=AC/RFC-
822=JJ#l#a#r#seismo.css.gov/@monet.berkeley.edu

où monet.berkeley.edu représente la passerelle de A vers B; C=UK, A=BT, P=AC la passerelle de B vers C et JJ@seismo.css.gov , l'adresse du destinataire. Cette dernière est écrite dans la syntaxe du réseau C.

Si la passerelle de A vers B n'applique pas la syntaxe atom-encoded, l'adresse est modifiée et donc incomprise dans le réseau cible.

RFC-822	X400	RFC-822
JJ#l#a#r#	JJ#l#a#r#	JJ#h#l#h#a#h#r#h#
seismo.css.gov	seismo.css.gov	seismo.css.gov

Si la passerelle RFC-822 X400 applique la syntaxe atom-encoded, l'adresse reste correcte dans le réseau cible.

RFC-822	X400	RFC-822
JJ#l#a#r#	JJ(a)	JJ#l#a#r#
seismo.css.gov	seismo.css.gov	seismo.css.gov

La transformation pour toute adresse 822P1 Recipient permet ainsi à un site d'adresser un réseau qui ne supporte pas le quoted-string sans que la passerelle le sache explicitement.

b) Correspondance entre 822-address et ORName.

L'étude de la correspondance entre 822-address et ORName se fonde sur la démarche suivante : une spécification de la transformation des adresses selon les restrictions adoptées par le document de RARE suivie d'une suppression de ces restrictions et terminée par une description de l'algorithme général.

1).Transformation des adresses selon Ruediger Grimm.

L'association RARE (Réseaux Associés pour la Recherche Européenne) fut fondée en juin 1986. Ces membres sont responsables pour la recherche sur l'infrastructure des réseaux. C'est un objectif de RARE d'établir une communication commune pour l'europe entière. Dans ce cadre, Ruediger Grimm a publié un article sur lequel est basé ce chapitre.[Grimm R.]

1.1. Hypothèses.

1.1.1. RFC-822.

Une adresse RFC-822 a la structure générale

local-part@sdom(n).sdom(n-1)...sdom(2).sdom(1).dom

où "sdom" remplace "subdomain" et "dom", "domain" avec la hiérarchie :

local-part<sdom(n)<sdom(n-1)<...<sdom(1)<dom

1.1.2. X400

Parmi les attributs d'un ORName, seuls sont choisis les éléments surName (S),givenName (GI), initials (I), Sequence of OrganizationalUnit (OU,OU,...), OrganizationName (O), PrivateDomainName (P), AdministrationDomainName (A), CountryName (C) avec la hiérarchie :

PersonalName < OU < OU <...< O < P < A < C

Ruediger Grimm fait l'hypothèse que l'usage des éléments DomainDefinedAttribute ne deviendra jamais une pratique courante. Cette supposition provient du profil international lui-même. En effet, Ruediger Grimm estime que toute installation qui ne veut pas s'isoler va devoir abandonner ses attributs non standards pour les attributs standards.

De ce fait, dans la majorité des cas, ces restrictions n'en sont plus.

1.2. Correspondance de base.

I.2.1.X400 vers RFC-822.

Les attributs standards sont encodés en

<PersonalName>@<OU>.<OU>.<O>.<P>.<A>.<C>

c.-à-d.

<PersonalName>	----->	local-part
<OU le plus bas>	----->	sdom(n)
<OU suivant>	----->	sdom(n-1)
.....		
<O>	----->	sdom(3)
<P>	----->	sdom(2)
<A>	----->	sdom(1)
<C>	----->	dom

Si un attribut standard est manquant, alors il est omis et l'attribut standard le suivant dans la hiérarchie prend sa place.

Ainsi une adresse X400 qui a une seule OrganizationalUnit et pas de OrganizationName est transformée en <S>@<OU>.<P>.<A>.<C>

De même l'adresse X400 S=schneider; OU=pcs1; OU=darmstadt; P=gmd; A=dbp; C=de est transformée en :

schneider@pcs1.darmstadt.gmd.dbp.de

1.2.2. RFC-822 vers X400.

L'adresse RFC-822 :

<local-part>@<sdom(n)>.<sdom(n-1)>...<sdom(2)>.<sdom(1)>.<dom>

est transformée en

<local-part>	----->	PersonalName
<sdom(n)>	----->	OU le plus bas
<sdom(n-1)>	----->	OU suivant
...		
<sdom(4)>	----->	OU le plus haut
<sdom(3)>	----->	OrganizationName O
<sdom(2)>	----->	PrivateDomainName P
<sdom(1)>	----->	AdministrationDomainName A
<dom>	----->	CountryName C

si n<2 alors P,O et OUs sont inutilisés

si n<3 alors O et OUs sont inutilisés

si n<4 alors OUs sont inutilisés

Ainsi

meyer@a2.utb.fgh.dbp.de

est transformé en

S=meyer; OU=a2; O=utb; P=fgh; A=dbp; C=de

De même

postmaster@ptt.at

est encodé en

S=postmaster; A=ptt; C=at

1.3. Correspondance entre local-part et PersonalName.

La correspondance entre local-part et PersonalName est basée sur la syntaxe :

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

given = 2*<ps-char ne contenant pas le caractère ".">

initial = ALPHA

surname = printablestring

1.3.1.X400 vers RFC-822.

Les attributs existants du PersonalName sont concaténés selon la syntaxe encoded-pn.

Ainsi GI=marshall; I=MT; S=Rose donne :

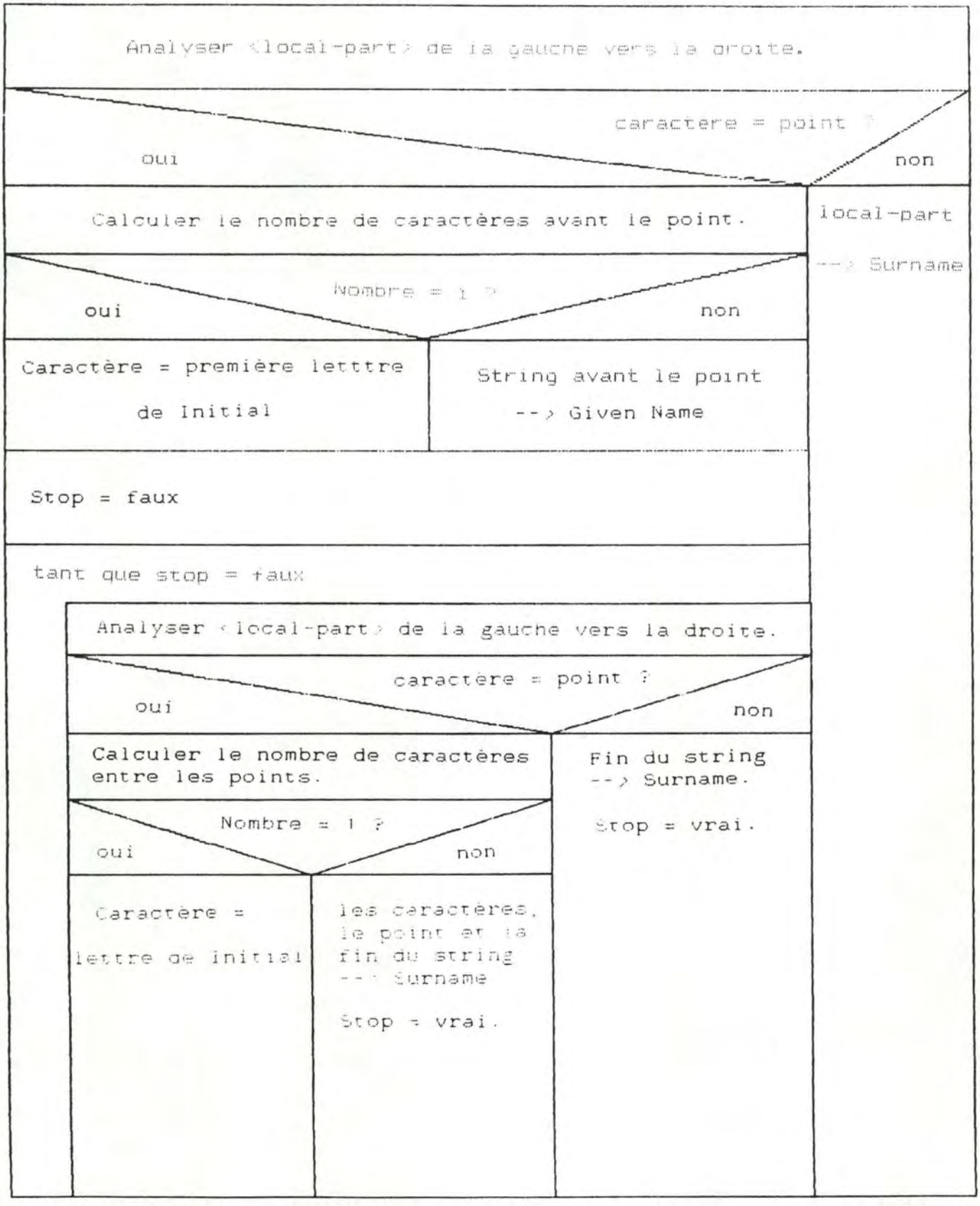
<local-part> = marshall.M.T.Rose

De même GI=marshall; I=; S=Rose donne :

<local-part> = marshall.Rose

1.3.2.RFC-822 vers X400.

La local-part est analysée en fonction de la syntaxe encoded-pn, selon l'algorithme suivant :


Analyser <local-part> de la gauche vers la droite.
caractère = point ?
oui
non
Calculer le nombre de caractères avant le point.
local-part
--> Surname
Nombre = 1 ?
oui
non
Caractère = première letttre de Initial
String avant le point
--> Given Name
Stop = faux
tant que stop = faux
Analyser <local-part> de la gauche vers la droite.
caractère = point ?
oui
non
Calculer le nombre de caractères entre les points.
Fin du string
--> Surname.
Stop = vrai.
Nombre = 1 ?
oui
non
Caractère = lettre de initial
les caractères, le point et la fin du string
--> Surname
Stop = vrai.

Exemples :

	<local-part>	PersonalName		
1)	Rose	GI=;	I=;	S=Rose
2)	M.Rose	GI=;	I=M;	S=Rose
3)	Marshall.Rose	GI=Marshall;	I=;	S=Rose
4)	Marshall.M.T.Rose	GI=Marshall;	I=MT;	S=Rose
5)	Marshall.M.Rose.Bs	GI=Marshall;	I=M;	S=Rose.Bs
6)	Marshall.MT.Rose	GI=Marshall;	I=;	S=MT.Rose

1.4. Correspondance de base contrôlée par les exceptions.

La correspondance de base définie au paragraphe précédent n'est pas entièrement symétrique et globale puisqu'il peut y avoir des absents.

En effet l'adresse X400 S=postmaster; OU=education; A=ptt; C=at donne comme représentation RFC-822:

postmaster@education.ptt.at.

Cette dernière adresse retraduite en X400 devient

S=postmaster; P=education; A=ptt; C=at.

De même l'adresse RFC-822 GRIMM@DDAGMD11.BITNET est transformée en une adresse illégale :

S=GRIMM; A=DDAGMD11; C=BITNET

car BITNET n'est pas un CountryName.

La correspondance de base nécessite une administration coopérative entre les mondes X400 et RFC-822 de façon à ce que domaines et attributs standards soient significatifs de chaque côté. De plus, pour cette correspondance, les adresses X400 ne peuvent outrepasser un attribut de la hiérarchie sans le risque de perdre la symétrie.

Pour surmonter ces restrictions, des tables de conversion sont définies aux passerelles. Elles renferment les adresses qui ne sont pas transformées par défaut.

1.4.1. Tables de conversion.

Deux tables sont définies; une pour chaque direction de la correspondance. Leur partie gauche spécifie les adresses sources, la partie droite les adresses cibles.

Une spécification d'adresse est vide ou doit comprendre au minimum le Top-domain pour le côté RFC-822 ou le CountryName pour le côté X400.

pour la représentation des éléments d'une table, nous adoptons le format :

entree 1*(SPACE) "#" sortie CRLF

où entree et sortie représente des adresse X400 et RFC-822 de syntaxe correcte.

exemple :

att.be #A=rtt; C=be

1.4.2. X400 vers RFC-822.

Etant donné une adresse d'un message passant du monde X400 vers RFC-822, la passerelle doit procéder comme suit :

- Lire les entrées sources de la table de conversion appropriée.
- Vérifier, pour chaque entrée, si elle apparaît intégralement dans une partie de l'adresse donnée.
- Si oui, choisir cette entrée si son degré d'appariement est le plus élevé.
- Dans tout autre cas, ne pas tenir compte de l'entrée.

Le degré d'appariement est le nombre d'attributs d'ordre plus élevés s'appariant avec l'adresse X400 à transformer.

Ainsi, considérons l'adresse X400 :

S=Frank; OU=pw; P=mpg; A=dbp; C=de

et les entrées

1) C=de; A=dbp
2) C=de; A=dbp; O=;
3) C=de; A=dbp; P=mpg;
4) C=de; A=dbp; P=fhg;

Seules les entrées 1, 2, 3 se retrouvent complètement dans l'adresse X400 et l'entrée 3 est choisie car elle possède le degré d'appariement le plus élevé. Elle a le plus grand nombre d'attributs d'ordre plus élevé (P est plus haut que O).

- Si aucune entrée n'apparaît dans l'adresse appliquer la correspondance de base.
- Sinon,
  - lire l'adresse cible correspondant à l'entrée choisie. Elle constitue la partie supérieure de l'adresse RFC-822.
  - Si l'entrée est identique à l'adresse , compléter la transformation par la correspondance PersonalName sur local-part.
  - Sinon mettre les attributs de l'adresse du message qui ne sont pas dans l'entrée source dans l'ordre naturel, les concaténer avec l'adresse cible en appliquant la correspondance de base et terminer la transformation par la correspondance PersonalName sur local-part.

Ainsi, étant donné la table :

BITNET #P=BITNET; A=DBP; C=DE

l'adresse GRIMM@DDAGMD11.BITNET est traduite en S=GRIMM; O=DDAGMD; P=BITNET; A=DBP; C=DE.

1.4.3. RFC-822 vers X400.

Etant donné une adresse d'un message passant du monde RFC-822 vers X400, la passerelle doit procéder comme suit :

- Lire les entrées sources de la table de conversion appropriée .
- Vérifier pour chaque entrée si elle apparaît intégralement dans l'adresse donnée en partant de la droite vers la gauche .
- Si oui, choisir cette entrée si son degré d'appariement est le plus élevé .
- Dans tout autre cas ne pas tenir compte de l'entrée.

Une entrée a un degré d'appariement plus élevé si elle a plus de sous-domaines s'appariant avec l'adresse donnée.

- Si aucune entrée n'apparaît dans l'adresse, appliquer la correspondance de base (exemple 1) .
- Sinon,
  - lire l'adresse cible correspondant à l'entrée choisie.
  - Si l'entrée est identique à l'adresse d'origine (exemple 2)
    compléter la transformation par la correspondance local-part sur PersonalName.
  - Sinon appliquer la correspondance de base au reste de l'adresse donnée en tenant compte des attributs déjà spécifiés par la table et terminer la transformation par la correspondance local-part sur PersonalName (exemple 3).

exemple :

supposons la table :

```
ac.be                      #P=ac; A=rtt; C=be

exemple1. Smith@ptt.at       -->  S=Smith; A=ptt; C=at
exemple2. Smith@ac.be        -->  S=Smith; P=ac; A=rtt; C=be
exemple3. Smith@fndp.ac.be   -->  S=Smith; O=fndp; P=ac;
                                   A=rtt; C=be
```

2) Généralisation du document de Ruediger Grimm.

2.1. Adresses X400 pures.

La transformation d'un OR/Name contenant des attributs autres que ceux spécifiés dans RARE s'effectue au moyen de la syntaxe std-ORName

```
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 et un ps-char sauf "/" et "=">

std-pair = "$"(ps-delim/ps-char)
```

Si le type est "registered-dd-type", il doit être enregistré au NIC.

Si le type est "PN", la valeur est interprétée selon la syntaxe encoded-pn.

L'information ORName est encodée selon l'algorithme :

1. Déterminer le maximum d'attributs qui a une spécification de domaine dans la table de conversion adéquate.
2. Si aucune correspondance n'est trouvée, la spécification du domaine est celle de la passerelle.
   Sinon, elle est celle de la table de conversion.
3. Mettre les composantes restantes dans la partie locale de l'adresse au moyen de la syntaxe std-orname.

2.2. Adresses RFC-822 pures.

Nous considérons ici, les adresses sous la forme route address-spec ainsi que les adresses RFC-822 référençant une boîte-aux-lettres situées sur un réseau RFC-822 et dont la partie domaine n'a pas de correspondant dans le monde X400.

L'adresse RFC-822 est intégrée dans la partie DomainDefinedAttribute de l'ORName avec un type correspondant à l'environnement.

Trois types sont définis :

1. le type "RFC-822". La valeur associée doit être interprétée dans le contexte de RFC-822 et de RFC-920.
2. le type "JNT-Mail". La valeur associée doit être interprétée dans le contexte du protocole JNT Mail (Joint Network Team).
3. le type "UUCP". La valeur associée doit être interprétée selon les contraintes du monde UUCP.

Les autres attributs de l'ORName identifient la passerelle.

Remarquons que l'algorithme de base contrôlé par les exceptions applique l'algorithme de base pour toutes les adresses qui n'ont pas de correspondants dans la table . De ce fait, les adresses RFC-822 référençant une boîte-aux-lettres situées sur un réseau RFC-822 et dont la partie domaine n'a pas de correspondant dans le monde X400 seront traduites par l'algorithme de base et donc erronées.

Dans l'algorithme général, nous ne tiendrons pas compte de ces adresses car la passerelle devrait faire la distinction entre les adresses RFC-822 pures et les adresses RFC-822 pour lesquelles l'algorithme de base se justifie.

Nous rejetons également à la passerelle les adresses 822-P1 Recipient qui ont la structure "route local-part@domain" car la partie "route" spécifie une liste d' hôtes à traverser qui ne peut être interprétée dans le monde X400.

2.3.Multiples correspondances.

La syntaxe std-orname est particulièrement adéquate pour spécifier une route à travers divers réseaux X400 et RFC-822.

L'utilisateur indique les adresses des différentes passerelles et l'adresse du destinataire dans le format du dernier réseau.

Ainsi, par l'adresse

```
C           = "UK"
A           = "BT"
P           = "AC"
"JNT-Mail"  = "/PN=Duval/DD.title=Manager/(a)Inria.PTT.FR"
```

le message est envoyé à la passerelle située en C=UK; A=BT; P=AC; puis à la passerelle déterminée par FR.PTT.Inria qui construit l'ORName :

C	=	"FR"
A	=	"PTT"
P	=	"Inria"
S	=	"Duval"
"Title"	=	"Manager"

3) Algorithmes généraux.

3.1.RFC-822 vers X400.

Etape 1.

1. Vérifier que la 822-address est de la forme
   local-part"@"domain.
   Sinon aller à l'étape 2.

2. Vérifier que le domaine est de la forme
   *(domain-syntax".") domain-syntax.
   Sinon aller à l'étape 2

3. Si le domaine n'identifie pas explicitement la passerelle, appliquer l'algorithme de RFC-822 vers X400 de la correspondance contrôlée par les exceptions.

4. Traduire la local-part en ASCII selon la syntaxe atom-encoded si
   A. le réseau source ne supporte pas le quoted-string
   ou
   B. l'adresse est un 822-P1 Recipient.

5. Traduire le résultat de 4 en printablestring selon la syntaxe ps-encoded.

6. Analyser le résultat de 5 suivant la syntaxe std-orname.

7. Si 6 échoue, analyser le résultat de 5 selon la syntaxe encoded-pn.

8. Si 7 échoue, aller à l'étape 2, sinon le résultat est un ensemble de paires type/valeur.

9. Vérifier que la valeur de chaque type est conforme à la syntaxe attribute-value, sinon aller à l'étape 2.

10. S'assurer que l'ensemble des attributs est conforme à la spécification d'un ORName, sinon aller à l'étape 2

11. Construire l'ORName selon sa représentation standard.

Etape 2

Cette étape est atteinte si l'adresse ne peut être encodée en une adresse X400 valide (1.2,1.8,1.9,1.10) ou si l'adresse est de la forme route local-part"@"domain.

Pour ce dernier cas,

1. Vérifier que l'adresse est de la forme route local-part"@"domain.

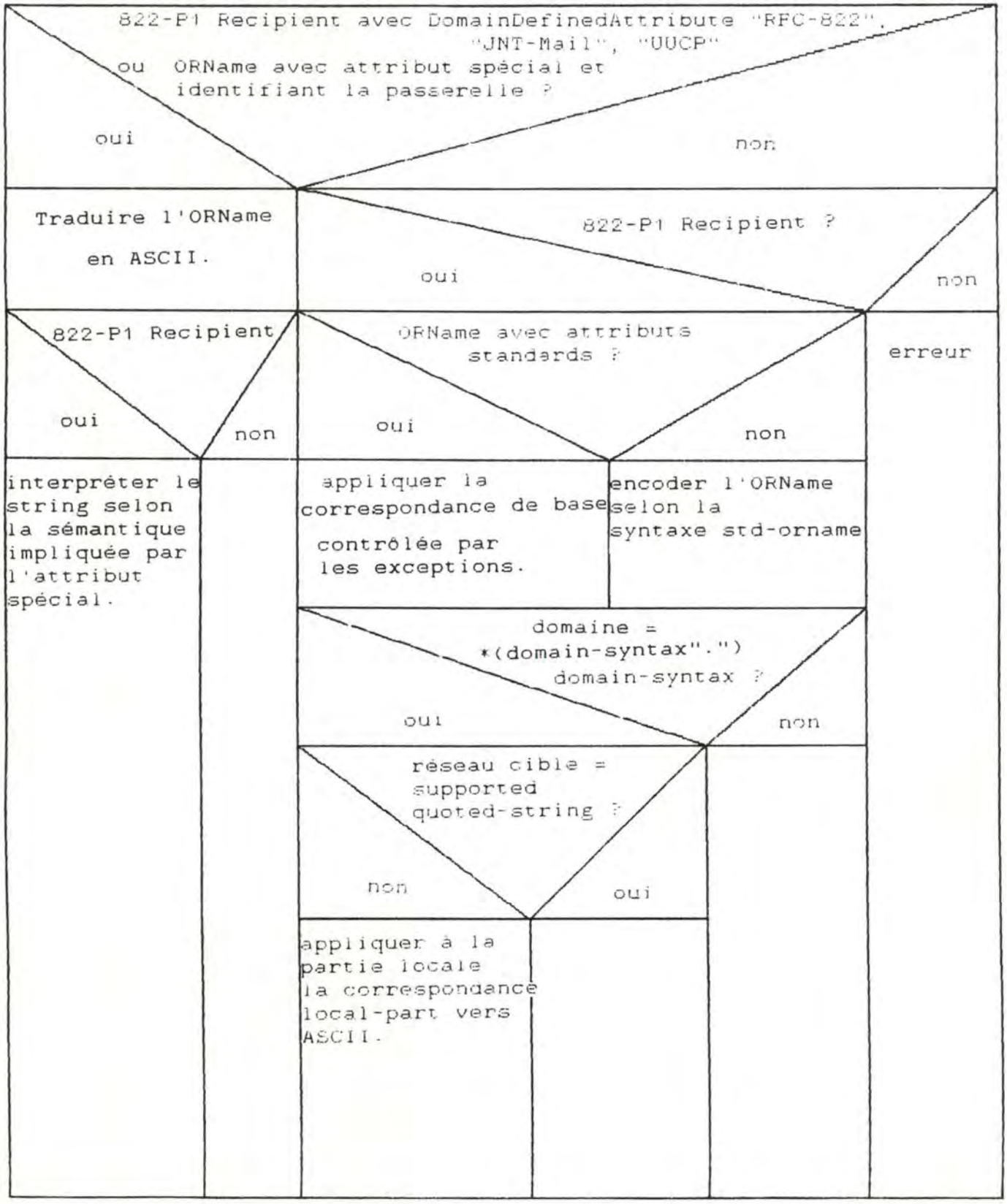
2. Si l'adresse est un 822 P1 Recipient, elle doit être rejetée.

3. Sinon, convertir l'adresse en printablestring selon la syntaxe ps-encoded.

4. Choisir le DomainDefinedAttribute ("RFC-822", "JNT-Mail", "UUCP") propre à la passerelle avec comme valeur l'adresse 822- address.

5. Construire le reste de l'ORName de façon à ce qu'il reçoive comme interprétation globale correcte, l'adresse de la passerelle..

3.2. X400 vers RFC-822.



CHAPITRE V : CORRESPONDANCE DES SERVICES.

L'interconnexion entre deux mondes de courrier électronique respectant des syntaxes distinctes est subdivisée en deux problèmes : la correspondance des adresses et la correspondance des services. La conversion d'une adresse de format RFC-822 en ORName et réciproquement a été présentée dans le chapitre IV et des algorithmes généraux y sont décrits. Le chapitre V définit la correspondance entre les éléments de protocole au sens X400 et les champs de la syntaxe RFC-822, pour fournir les services d'un monde indépendamment du passage par la passerelle.

Cette interconnexion utilise les spécifications des algorithmes généraux de conversion entre une adresse RFC-822 et un ORName : toute correspondance entre un élément de protocole comprenant un ORName et un champ d'adresse est donc supposée acquise.

Une telle passerelle fait le lien entre une entité MTAE du MTS et le système de transfert d'un RFC-822 message : il est inutile de présenter le protocole P3 régissant l'interaction entre les entités SDE et MTAE, puisqu'il n'interviennt aucunement dans l'interconnexion de mondes X400 et RFC-822.

La première partie de ce chapitre présente les services offerts aux utilisateurs X400 et définit les protocoles P1 et P2; la seconde décrit les champs d'un RFC-822 message et la troisième spécifie la correspondance les éléments de protocole X400 et les champs de la syntaxe RFC-822.

I) Description des services dans les Recommandations X400.

La première partie du chapitre III a montré que les normes X400 sont le fruit de la modélisation abstraite d'un système de corrier électronique et de la subdivision de la couche APPLICATION en deux sous-couches. Le but de cette partie est de citer et d'expliquer les services offerts par la couche MTL à la couche UAL (MTS = Message Transfer Service), elle-même offrant des services aux utilisateurs; les normes X400 n'ont décrit pour le moment qu'un seul type de services offerts par la couche UAL : le service de messagerie interpersonnelle (IPMS = Interpersonal Messaging Service) .

Il est à rappeler que la description des services offerts par une entité détaille les fonctionnalités de cette entité tandis que l'utilisation concrète de services s'effectue par l'appel de primitives et par l'affectation adéquate des paramètres. Certains éléments de service sont fournis localement et ne dépendent pas d'une communication entre plusieurs entités. D'autres nécessitent une coopération entre éléments actifs d'une couche : leur implémentation nécessite l'affectation des champs adéquats dans le cadre des unités de données de protocole que s'échangent les entités paires.

Quatre étapes sont indispensables pour l'exécution d'une primitive dont la fonctionnalité est de répondre à une demande d'un élément de service nécessitant une communication entre deux entités paires (cfr figure V.1). Une entité de niveau n d'un système A désire utiliser un service SERV offert par le niveau n-1, pour communiquer avec une entité paire du système B. Pour ce faire, elle appelle la primitive qui va demander l'exécution du service en passant comme paramètre les informations nécessaires (INFO) : SERV.request(INFO). Par l'interconnexion des deux systèmes, cette demande est transmise à l'entité de niveau n-1 du système B qui signale à l'entité n, l'exécution d'un service souhaité par une entité n du système A, via la primitive SERV.indication(INFO). L'entité de niveau n

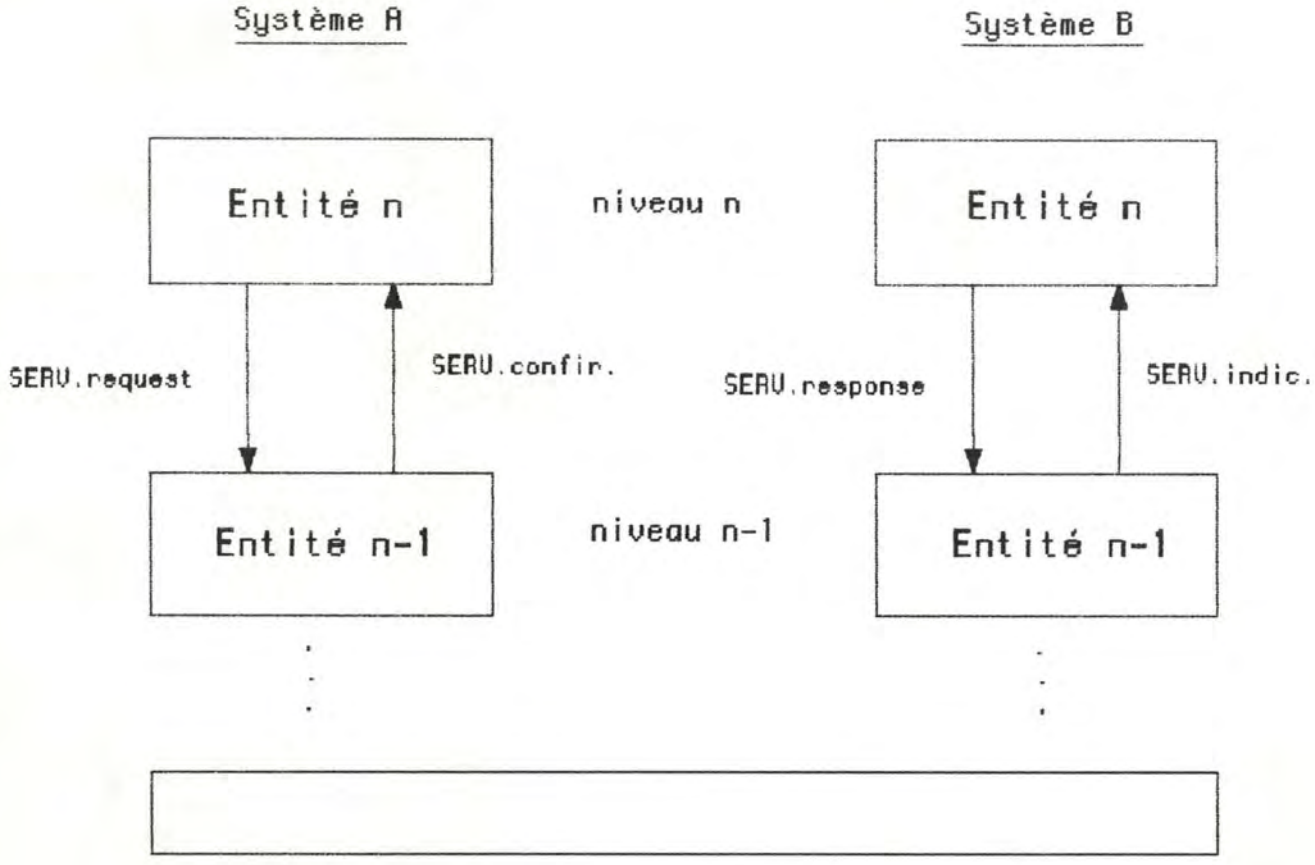


FIGURE V.1 : EXECUTION D'UNE PRIMITIVE EN QUATRE TEMPS

du système B valide ou pas l'exécution du service et renvoie la réponse en appelant la primitive SERV.response avec deux arguments : l'acceptation ou le refus, la raison du refus s'il y a. Par le réseau, l'exécution du service est confirmée ou non à l'entité n du système A par la primitive SERV.confirmation.

Dans le cadre d'une messagerie interpersonnelle, une personne désire envoyer un message à un ou plusieurs destinataires. Pour ce faire, elle accède à l'UAE dont elle dispose et, par une primitive et ses paramètres, lui communique le contenu du message, le(s) destinataire(s) et les services souhaités. L'entité UAE construit un PDU (appelé UAPDU) en y affectant les valeurs des paramètres de la primitive dans les champs adéquats. Cet UAPDU dont la structure est normalisée par le protocole P2 contient dès lors toutes les informations nécessaires au transfert vers le(s) destinataire(s) : le contenu du message, le(s) destinataire(s) et les services demandés par l'utilisateur et dont la réalisation nécessite la collaboration de(s) UAE destinataire(s). Le UAE expéditeur appelle ensuite la primitive de soumission (SUBMIT) offerte par le MTAE connecté : l'UAPDU est passé comme paramètre de cette primitive de même que les services du MTS souhaités. Le MTAE construit un PDU défini par le protocole P1 (appelé MPDU), y encapsule l' UAPDU et y affecte les informations de services dans les champs adéquats. Ce MPDU est dès lors confié au MTS (Message Transfer System) qui se charge du transfert vers les MTAE connectés aux UAE destinataires, via des MTAE relais. Du MPDU transféré, le MTAE destinataire extrait l'UAPDU qu'il livre au UAE, et les informations sur les services souhaités. De même, l'UAE dépouille l'UAPDU et y sépare le message et les informations de services.

Ci-dessous seront cités et explicités les éléments de services de transfert de messages (MTS) et de messagerie interpersonnelle (IPMS). Cette description comporte deux parties : la première définit la sémantique des éléments de service et la deuxième décrit les PDU correspondants.

a) Description des services de transfert de messages (MTS).

A) Les services offerts par le MTS.

Le système de transfert de messages se compose d'entités agent de transfert de message (MTAE) : les MTAE s'échangent un message soumis au MTS par un UAE expéditeur jusqu'à ce que le message atteigne le MTAE auquel est connecté l'UAE destinataire. Le MTS offre un service de transfert de messages avec un routage incrémental : un MTAE recevant un message analyse la destination et décide d'envoyer le message à un MTAE relais sans connaître le chemin au-delà, vers le MTAE destinataire. Trois types d'interaction entre un UAE et son MTAE permettent l'échange de messages :

- **l'interaction de soumission** permet à un UAE expéditeur de transférer à un MTAE le contenu d'un message et les informations dont le MTS a besoin pour véhiculer le message et fournir les éléments de service demandés. La soumission est réalisée par l'appel de la primitive SUBMIT.

- **l'interaction de livraison** donne la possibilité à un MTAE de remettre un message à un UAE destinataire. La livraison est effectuée par l'appel de la primitive DELIVER.

- **l'interaction de relais** offre l'échange de messages entre MTAE et s'établit par la soumission à la couche inférieure du modèle OSI.

Les éléments de service du MTS sont classés en services de base et en services optionnels. Les services de base de transfert de messages permettent d'envoyer et de recevoir des messages. Tout UAE a la possibilité de spécifier le type d'information encodée qui peut apparaître dans les messages qui lui sont livrés (type télex, télétex, voix, image, IA5string, Simple Formattable Document, ...). Un UAE expéditeur appelle la primitive SUBMIT pour qu'un message soit transféré à un ou plusieurs destinataires. Il peut également, lors de la soumission d'un message, spécifier le type de contenu qui identifie la convention qui régit la structure du contenu (P2 pour la messagerie interpersonnelle). Le MTS assigne une référence univoque à chaque copie de messages qu'il a dû produire pour livrer à tous les destinataires. Si un message ne peut pas être livré, le UAE expéditeur en est informé. Pour chaque message livré, les paramètres de la primitive DELIVER spécifient à l'UAE destinataire, l'heure d'expédition et de livraison, le type de contenu, le type de l'information encodée à l'expédition, une indication des conversions qui ont été effectuées et le type d'information encodée résultant.

En plus des services de base, le MTS offre des facilités à son utilisateur fournissant des éléments de service à option : pour certains de ces services, le choix de leur utilisation s'effectue pour chaque message tandis que pour d'autres, il se fait pour une période de temps déterminée.

1. Les services de transfert de message de base.

Pour transférer un message, le UAE expéditeur soumet le message au MTS avec le nom ou l'adresse du(es) destinataire(s). Le MTS a terminé la livraison quand le message est transmis au(x) UAE destinataire(s) avec l'identité de l'expéditeur. Ces fonctionnalités du MTS nécessitent l'offre de services de base qui sont :

1.1. L'identification du message.

En cours de routage, le MTS produit des copies du message suivant les différentes destinations et identifie chaque copie de manière univoque. Les UAE et le MTS utilisent cet identificateur pour désigner un message précédemment envoyé pour d'autres éléments de service tels que le rapport de livraison ou de non-livraison.

1.2. Le rapport de non-livraison.:

Le MTS avertit un UAE expéditeur quand le message envoyé n'a pas été livré à un destinataire souhaité. La raison pour laquelle le message n'a pas été livré est incluse dans le rapport. Dans le cas d'un message multi-destinataire, un rapport de non-livraison existe pour tout destinataire spécifié n'ayant pas reçu le message.

1.3. Les types d'informations encodées possibles.

Par cet élément de service, un UAE informe le MTS des types d'informations encodées qui peuvent lui être livrés.

1.4. L'indication des types d'informations encodées à la soumission.

Le UAE expéditeur spécifie au MTS les types d'informations encodées du message envoyé. Quand le message est livré, le MTS signale à l'UAE destinataire les types d'informations du message à l'origine (avant toute conversion).

1.5. L'indication de conversion.

Le MTS indique à un UAE destinataire qu'il a effectué une conversion de type d'information encodée sur le message livré et spécifie les types résultants.

1.6. L'indication de l'instant de soumission.

Le MTS signale à l'UAE expéditeur et l'UAE destinataire la date et l'heure de l'envoi du message au MTS.

1.7. L'indication de l'instant de livraison.

Le MTS indique à l'UAE destinataire la date et l'heure de livraison du message.

1.8. La gestion de l'accès.

Cet élément de service permet à un UAE d'établir l'accès à un MTAE et de gérer les informations liées à cette association.

1.9. L'indication du type de contenu.

Par cet élément de service, l'UAE expéditeur signale le type de contenu pour chaque message envoyé (P2).

2. Les services de transfert de messages à option.

Ces éléments de service sont classés en trois groupes.

2.1. Les services de soumission et de livraison.

Les services suivants peuvent être choisis :

2.1.1. Le choix du degré de livraison.

Le UAE expéditeur peut demander que le transfert du message dans le MTS se fasse de manière urgente, normale ou non-urgente.

2.1.2. La livraison multi-destinataire.

Le UAE expéditeur peut spécifier que le message soumis au MTS doit être livré à plus d'un destinataire.

2.1.3. La divulgation des autres destinataires.

Un UAE expéditeur accepte que, dans le cas où il a envoyé un message pour plusieurs destinataires, le MTS révèle à chaque UAE destinataire l'identité des autres destinataires.

2.1.4. La permission de livrer à un destinataire suppléant.

Un UAE expéditeur peut spécifier que le message envoyé peut être livré à un autre destinataire que celui souhaité à l'origine, dans le même MD (Management Domain). Deux conditions doivent être remplies pour la nécessité et l'exécution correcte de cet élément de service : il faut que les attributs de l'ORName destinataire n'identifient pas un UAE R2EL mais désigne un MD réel; de plus, il faut que ce MD fournisse l'élément de service d'assignation d'un destinataire suppléant et qu'un UAE ait été désigné comme tel pour recevoir ces messages.

2.1.5. La livraison différée.

Un UAE expéditeur peut demander au MTS que le message envoyé ne soit pas livré avant une date et une heure spécifiées.

12.1.6. Le rapport de livraison.

Un UAE expéditeur peut demander qu'un rapport lui soit envoyé quand un message soumis au MTS a été livré à l'UAE destinataire. Ce rapport de livraison désigne le message livré par son identificateur et précise la date et l'heure de livraison.

2.1.7. L'interdiction de rapport de non-livraisoN.

Un UAE expéditeur peut demander que le MTS ne lui renvoie pas un rapport de non-livraison dans le cas où le message soumis n'a pas pu être livré.

2.1.8. La suppression de livraison différée.

Un UAE expéditeur peut demander au MTS d'interdire la livraison d'un message dont la soumission a été acceptée mais dont la livraison était différée.

2.1.9. Le retour de contenu.

Un UAE expéditeur peut demander que le contenu d'un message lui soit renvoyé avec le rapport de non-livraison. Cela ne sera pas fait si une conversion a été effectuée sur le contenu du message.

2.2. Les services de conversion.

Les éléments suivants de conversion de type d'informations encodées peuvent être choisis :

2.2.1. L'interdiction de conversion.

Un UAE expéditeur peut demander au MTS qu'aucune conversion de type d'informations encodées ne soit exécutée sur le message soumis.

2.2.2. La conversion implicite.

Cet élément de service permet à un UAE de demander que le MTS fasse toute conversion qui soit adéquate sur les messages envoyés et ce, pendant une période de temps spécifiée.

2.2.3. La conversion explicite.

Un UAE expéditeur peut demander au MTS d'exécuter une conversion spécifique sur le message soumis.

2.3. Le service de sonde.

Le service de sonde permet de simuler la soumission d'un message et de générer un rapport de livraison ou de non-livraison indiquant si un message avec les mêmes informations de soumission pourrait être livré aux UAE destinataires spécifiés.

2.4. Les services de statut.

2.4.1. L'assignation d'un destinataire suppléant.

Cet élément de service désigne, au sein d'un MD, un UAE pour recevoir des messages pour lesquels les attributs du destinataire ne correspondent pas à un UAE connu. Une organisation peut dès lors décider qu'un UAE reçoit tous les messages dont l'ORName destinataire ne désigne pas univoquement un UAE, pour distribuer manuellement ces messages.

2.4.2. L'attente pour la livraison.

Un UAE destinataire peut demander au MTS de garder les messages ou les rapports à lui livrer pendant un certain temps. Le UAE peut indiquer qu'il est dans l'impossibilité de prendre livraison des messages ou des notifications et le moment où il sera prêt à les recevoir.

Il est à noter que presque tous ces éléments de service sont choisis par le UAE utilisateur pour chaque message envoyé sauf la conversion implicite et les services de statut que le UAE utilise pour une période de temps déterminée.

3. tableau récapitulatif des MT services :

1. services de transfert de message de base :

1.1. identification du message.
1.2. rapport de non-livraison.
1.3. types d'informations encodées possibles.
1.4. indication des types d'informations encodées à l'expédition.
1.5. indication de conversion.
1.6. indication de l'instant de soumission.
1.7. indication de l'instant de livraison.
1.8. gestion de l'accès.
1.9. indication du type de contenu.

2. services de transfert de message à option :

2.1. services de soumission et de livraison :

2.1.1. choix du degré de livraison.
2.1.2. livraison multidestinataire.
2.1.3. divulgation des autres destinataires.
2.1.4. permission de livrer à un destinataire redirigé.
2.1.5. livraison différée.
2.1.6. rapport de livraison.
2.1.7. interdiction de rapporter la non-livraison.
2.1.8. suppression de livraison différée.
2.1.9. retour de contenu.

2.2. services de conversion :

2.2.1. interdiction de conversion.
2.2.2. conversion implicite.
2.2.3. conversion explicite.

2.3. service de sonde.

2.4. services de statut :

2.4.1. assignation du destinataire de redirection.
2.4.2. attente pour la livraison.

B) Le protocole de transfert de messages (P1).

Certains services cités ci-dessus n'exigent que l'exécution de fonctions locales à un MTAE et ne dépendent pas d'une communication entre MTAE : la gestion de l'accès, les types d'informations encodées possibles, la suppression du délai de livraison et les trois éléments de service choisis pour une période de temps déterminée, c'est-à-dire la conversion implicite, l'assignation d'un destinataire de redirection et l'attente pour la livraison. Par contre, l'exécution des autres éléments de service nécessitent une parfaite coopération entre MTAE et exige la normalisation des structures d'informations à échanger pour une bonne compréhension. Une telle communication entre MTAE est possible grâce à P1, le protocole de transfert de messages.

Les éléments de protocole P1 sont appelés les unités de données de protocole de message (Message Protocol Data Unit = MPDU) et sont classés en deux groupes : les MPDU utilisateurs (User MPDU = UMPDU) et les MPDU de service (Service MPDU = SMPDU). Cette distinction entre MPDU permet une identification des fonctions pour lesquelles la coopération entre MTAE est nécessaire : les UMPDU contiennent les messages envoyés par un UAE expéditeur pour les transférer et les livrer à un autre UAE tandis que les SMPDU sont utilisés pour transmettre, entre MTAE, de l'information sur les messages : le MPDU de rapport de non livraison et le MPDU de service de sonde. Le premier, le Delivery Report MPDU, transporte un rapport de non livraison de messages vers le MTAE dont dépend l'UAE qui a expédié le message et le second, le Probe MPDU, contient les informations d'une soumission de message simulée dans le but de savoir si un message soumis avec les mêmes caractéristiques sera livré ou non.

Le protocole P1 définit de manière précise la structure d'un MPDU en différents champs qui contiennent les informations nécessaires à la collaboration entre MTAE : le contenu d'un message, les informations pour le transport, les demandes d'éléments de service, les réponses aux demandes de service.

La suite de cette partie décrira les champs des trois types de MPDU et montrera de manière précise où les éléments de service cités précédemment se retrouvent. La Recommandation X409 spécifie la syntaxe et les notations utilisées pour décrire les protocoles.

Tout MPDU se compose de deux parties : l'enveloppe et le contenu : l'enveloppe contient les informations que le MTS demande pour diriger le MPDU vers sa destination; le contenu est l'information à transférer.

1. Description d'un UMPDU.

Le contenu d'un UMPDU est un message que l'utilisateur souhaite transmettre et qui ne sera modifié que si une conversion de type est demandée. L'enveloppe contient onze champs qui contiennent l'information à transférer, nécessaire à l'exécution des éléments de service souhaités (cfr annexe 1) :

UMPDU enveloppe :	MT services correspondants :
MPDUidentifier	1.1. identification du message
OriginatorORname	
OriginalEncoded-InformationTypes	1.4. indication des types d'info. enc. à l'expédition
Contenttype	1.9. indication du type de contenu : P2
UAcontentid	
Priority	2.1.1. choix du degré de livraison
Permessageflag : DiscloseRecipients ConversionProhibited AlternateRecipient-Allowed ContentReturnRequest	 2.1.3. divulgation des autres destinataires 2.2.1. interdiction de conversion 2.1.4. permission de livrer à un destinataire suppléant 2.1.9. retour du contenu
DeferredDelivery	2.1.5. livraison différée
PerDomainBilateralInfo	

RecipientInfo : (pour tout destinataire)	2.1.2. livraison multidestinataire
RecipientORname	
ExtensionIdentifier	
PerRecipientFlag :	
ResponsabilityFlag	
ReportRequest	1.2. et 2.1.6. rapport de livraison ou de non-livraison
UserReportRequest	2.1.7. interdiction de rapporter la non-livraison
ExplicitConversion	2.2.3. conversion explicite

TraceInformation : (pour chaque domaine) GlobalDomain-Identifier DomainSuppliedInfo : Arrival	1.6. indication de l'instant de soumission
Defined	
Action	
Converted-InformationTypes	1.5. indication de conversion
PreviousGlobal-DomainIdent	

Tous les services nécessitant une collaboration entre MTAE ont un champ prévu dans le MPDU pour que les informations à communiquer soient repérées par le MTAE destinataire. Huit éléments de service offerts par le MTS ne nécessitent pas de champ dans le MPDU : ces services ne dépendent pas d'une communication entre MTAE mais caractérisent les liens entre un UAE et son MTAE :

- la gestion de l'accès,
- les types d'informations encodées possibles,
- la suppression de la livraison différée,
- la conversion implicite,
- l'assignation d'un destinataire suppléant,
- l'attente pour la livraison.

Le service de sonde et d'indication du moment de livraison concerne les deux autres types de MPDU.

2. Description du Delivery Report MPDU.

Le Delivery Report MPDU transfère un rapport de livraison ou de non-livraison vers l'expéditeur d'un message. Il est composé d'une enveloppe et d'un contenu. L'enveloppe contient les informations nécessaires au routage de ce MPDU dans le MTS : l'identificateur, l'expéditeur du message dont on notifie la livraison, et la trace de ce MPDU dans le MTS. Le contenu se compose de six champs (cfr annexe 1) dont l'identificateur du UMPDU qui transporte le message envoyé, le retour du contenu du message si ce service a été demandé, et la description de la livraison : la date et l'heure de livraison ou la raison de non-livraison.

3. Description du Probe MPDU

Le Probe MPDU contient la demande d'information sur la possibilité de livraison d'un message vers un ou plusieurs destinataires. Il ne se compose que d'une enveloppe qui contient les informations sur les éléments de service souhaités et sur les destinataires. Le Probe MPDU est transféré vers un MTAE destinataire qui renvoie un rapport de livraison ou de non-livraison (cfr annexe 1).

b) Description des services de messagerie interpersonnelle (IPMS).

A) Les services offerts par le IPMS (IPM System).

Dans le cadre d'une messagerie interpersonnelle, les UAE qui coopèrent pour permettre aux utilisateurs de s'échanger des messages, appelés les IP-messages, forment la classe des IPM-UAE.

Pour échanger un IP-message, un utilisateur compose son message et identifie le(s) destinataire(s) en dialoguant avec un IPM-UAE. Le IP-message est alors envoyé par l'IPM-UAE expéditeur vers l'IPM-UAE destinataire via le système de transfert de message (MTS). Si le IP-message est livré avec succès à l' IPM-UAE de l'utilisateur destinataire, celui-ci pourra lire le message. Un IPM-UAE offre à l'utilisateur l'ensemble de services de base de l'IPMS qui comprend les services de base du MTS ainsi que les services à option de l'IPMS incluant également les services à option du MTS.

De plus, tout IPM-UAE peut fournir des fonctions d'aide dans l'édition et dans la gestion des messages (éditeur et traitement de texte, gestionnaire de base de données pour enregistrer les messages envoyés et reçus). Ces services d'aide pour l'utilisateur sont des fonctions locales à un IPM-UAE, ne nécessitant pas de communication entre entités, et ne sont donc pas réglementés dans les Recommandations.

1. les services de messagerie interpersonnelle de base.

1.1. MT services de base.

1.2. L'identification du IP-message.

Cet élément de service permet d'identifier les IP-messages.

1.3. La nature du corps du message.

Cet élément de service permet de déterminer la nature des différentes parties du corps du IP-message (IA5Text, Teletex, G3Fac, ...).

2. Les MT services de soumission, de livraison, de conversion, de sonde et de statut.

Ces éléments de service permettent aux utilisateurs de demander à leur IPM UAE d'utiliser les services de soumission, de livraison, de conversion, de sonde et de statut du MTS.

3. Les services de messagerie interpersonnelle à option

Les éléments de service suivants peuvent être choisis par l'expéditeur d'un IP-message :

3.1. La désignation des destinataires primaires et secondaires.

Cet élément de service permet à l'expéditeur de fournir le nom d'utilisateur(s) désigné(s) comme destinataire primaire ou secondaire. Chacun peut dès lors déterminé dans quelle catégorie de destinataire il est spécifié. La distinction exacte entre les deux catégories n'est pas décrite dans les Recommandations. On peut considérer que les primaires sont les destinataires pour qui le message existe, dont on attend souvent une réponse, tandis que les secondaires reçoivent des copies de message pour information seulement.

3.2. la désignation de destinataire(s) non dévoilé(s).

Par cet élément de service, l'expéditeur d'un IP-message peut désigner un ou plusieurs destinataires supplémentaires dont l'identité ne sera pas divulgués aux destinataires primaires ou secondaires.

3.3. L'avis de non-réception.

Cet élément de service donne la possibilité à l'expéditeur de demander que lui soit envoyé un avis si le IP-message n'a pas été reçu par le destinataire souhaité. Si la destination du message est multiple, ce service est spécifié pour chaque destinataire. La raison pour laquelle le message n'a pas été reçu est incluse dans l'avis.

3.4. L'avis de réception.

Cet élément de service permet à l'expéditeur de demander qu'une notification lui soit envoyée pour signaler la réception ou la non-réception du message.

3.5. L'indication de transfert automatique vers un autre destinataire.

Certains utilisateurs peuvent demander de paramétrer leur IPM-UAE pour que les IP-messages lui étant destinés soient transférés automatiquement vers un autre IPM-UAE désigné. Cet élément de service permet à cet IPM-UAE désigné de savoir qu'un corps du IP-message reçu contient un message qui a été transféré automatiquement de son IPM-UAE destinataire.

3.6. La désignation de l'expéditeur.

Cet élément de service donne l'identité de l'expéditeur du IP-message à son destinataire.

3.7. La désignation des utilisateurs autorisant l'envoi du message.

Cet élément de service donne la possibilité à l'expéditeur de fournir le nom d'une ou de plusieurs personnes qui ont autorisé l'envoi du message.

3.8. L'indication de la date d'expiration.

Par cet élément de service, l'expéditeur peut indiquer au destinataire la date et l'heure après laquelle il considère que le IP-message n'est plus valide.

3.9. La désignation des liens entre messages.

L'expéditeur peut ainsi associer au IP-message envoyé les identificateurs d'un ou de plusieurs autres IP-messages auxquels il est fait référence, ce qui permet à l'IPM UAE destinataire de situer le message reçu par rapport à d'autres.

3.10. L'indication de l'importance.

L'expéditeur a la possibilité d'indiquer au destinataire son évaluation sur l'importance du IP-message qu'il a envoyé. Trois niveaux sont définis : bas, normal, haut.

3.11. L'indication d'annulation.

Cet élément de service donne la possibilité à l'expéditeur d'indiquer qu'un ou plusieurs IP-messages qu'il a envoyés précédemment sont à annuler, ce qui permet au IPM-UAE destinataire de les supprimer.

3.12. L'indication du degré de confidentialité.

L'expéditeur peut déterminer la sécurité qu'il exige à la réception du IP-message.

3.13. La désignation du sujet.

L'expéditeur a la possibilité d'indiquer au(x) destinataire(s) le sujet du IP-message envoyé.

3.14. L'indication de réponse.

Cet élément de service permet à l'expéditeur d'un IP-message d'indiquer au(x) destinataire(s) que ce message a été envoyé en réponse à un autre.

3.15. L'indication de demande de réponse.

Par cet élément de service, l'expéditeur peut demander que le destinataire envoie un IP-message en réponse au message qui contient la demande.

3.16. L'indication pour un IP-message transféré.

Un IPM-UAE peut transférer un IP-message qui lui était destiné; dans ce cas, cet élément de service permet d'indiquer qu'une partie du corps d'un IP-message contient un autre IP-message transféré.

3.17. L'indication d'encryptage d'une partie du corps.

Cet élément de service permet à l'expéditeur d'indiquer au destinataire que telle partie du corps du IP-message envoyé a été encryptée pour empêcher toute lecture et modification non autorisées. Cet élément de service sera utilisé par le destinataire pour déterminer quelle partie doit être décryptée.

3.18. L'indication de partitionnement du corps du IP-message.

Cet élément de service permet à un expéditeur d'envoyer un IP-message avec un corps composé de plusieurs parties de type différent (IA5Text, G3Fac, Teletex, ...). La nature ou le type de chaque partie est transportée dans le IP-message.

4. tableau récapitulatif des IPM services :

1. Services de base :

   1.1. MT services de base.
   1.2. Identification du IP-message.
   1.3. Type du corps du IP-message.

2. MT services à option.

3. Services de messagerie interpersonnelle à option :

   3.1. Désignation des destinataires primaires et secondaires.
   3.2. Désignation de destinataires non dévoilés.
   3.3. Avis de non-réception.
   3.4. Avis de réception.
   3.5. Indication de transfert automatique vers un autre destinataire.
   3.6. Désignation de l'expéditeur.
   3.7. Désignation des utilisateurs autorisant l'envoi du IP-message.
   3.8. Indication de la date d'expiration.
   3.9. Désignation des liens entre messages.
   3.10. Indication de l'importance.
   3.11. Indication de l'annulation.
   3.12. Indication du degré de confidentialité.
   3.13. Désignation du sujet.
   3.14. Indication de réponse.
   3.15. Indication de demande de réponse.
   3.16. Indication pour un IP-message transféré.
   3.17. Indication d'encryptage d'une partie de corps.
   3.18. Indication de partitionnement du corps du IP-message.

B) Le protocole de messagerie interpersonnelle (P2)

Tous les éléments de service de l'IPMS dépendent d'une communication entre IPM UAE, sauf les huit services du MTS qui sont des moyens de connection et de définition d'un MTAE. Cette collaboration entre IPM UAE nécessite une normalisation de la structure des informations à échanger; une telle normalisation est décrite dans le protocole de messagerie interpersonnelle, le protocole P2.

Les éléments de protocole P2 sont appelés les unités de données de protocole d'agent utilisateur (User Agent Protocol Data Unit = UAPDU). Deux types de UAPDU sont définis : les UAPDU de message (IP-message UAPDU = IM-UAPDU) et les UAPDU de rapport de statut (IPM-Status Report = SR-UAPDU).

Les IM-UAPDU contiennent le IP-message à échanger entre l'expéditeur et le destinataire, et sont composés d'un en-tête et d'un corps. L'en-tête comprend les informations nécessaires à l'échange du message et à l'exécution des services souhaités par l'utilisateur tandis que le corps est l'information que l'expéditeur désire communiquer. Les normes X400 permettent l'échange d'informations sous différentes formes : textes, voix, images graphiques. Un SR-UAPDU contient les avis de réception ou de non-réception d'un IP-message précédemment envoyé.

La suite de cette partie décrira les différents champs des IM-UAPDU et des SR-UAPDU, et précisera la correspondance avec les éléments de service du IPMS. La description des UAPDU utilise les mêmes règles de syntaxe et notations que celles utilisées pour décrire les MPDU. (cfr Recommandation X409).

1. description d'un IM-UAPDU.

Un IM-UAPDU dont la définition précise est présentée en annexe 1 se compose de l'en-tête et du corps :

- l'en-tête est un ensemble de seize champs dont chacun est utilisé pour fournir un ou plusieurs éléments de services.

- le corps est composé d'une ou de plusieurs parties qui peuvent contenir des informations de type différent; chaque partie a une indication du type d'informations qu'elle communique.

Composantes de l'en-tête d'un IM-UAPDU	IPM éléments de service correspondants
IPMessageId	1.2. identification du IP-message
Originator	3.5. désignation de l'expéditeur
AuthorizingUsers	3.6. désignation des utilisateurs autorisant l'envoi du IP-message
PrimaryRecipients **CopyRecipients** par destinataire	3.7. désignation des destinataires primaires et secondaires 3.15. indication de demande de réponse 3.3. avis de réception 3.2. avis de non-réception
BlindCopyRecipients par destinataire	3.1. désignation de destinataires non dévoilés 3.15. indication de demande de réponse 3.3. avis de réception 3.2. avis de non-réception
InReplyTo	3.14. indication de réponse

Obsoletes	3.11. indication de l'annulation
CrossReferences	3.9. désignation des liens entre messages
Subject	3.13. désignation du sujet
ExpiryDate	3.8. indication de la date d'expiration
ReplyBy **ReplyToUsers**	3.15. indication de demande de réponse
Importance	3.10. indication de l'importance
Sensitivity	3.12. indication du degré de confidentialité
Autoforwarded	3.4. indication de transfert automatique vers un autre destinataire

Corps d'un IM-UAPDU	IPM éléments de service correspondants
Body ::= SEQUENCE OF BodyPart	3.18. indication de partitionnement du corps du IP-message
BodyPart ::= CHOICE [	1.3. type du corps du IP-message
IA5Text, TLX, Voice, G3Fax, TIFO, TTX, VideoTex, NationallyDefined, Encrypted,	3.17. indication d'encryptage d'une partie de corps
ForwardedIPMessage,	3.16. indication pour un IP-message transféré
SFD, TIF1]	

2. description d'un SR-UAPDU.

Le SR-UAPDU est utilisé pour renvoyer un avis de réception ou de non-réception d'un IP-message vers son expéditeur. Différents champs y sont définis pour contenir des informations telles que l'identificateur du IP-message concerné, la raison de la non-réception de cet IM-UAPDU ou la date et l'heure de la réception, le type d'informations encodées après conversion, le destinataire spécifié à l'origine et le destinataire qui a reçu réellement le message (destinataire redirigé ou destinataire recevant automatiquement les messages d'un autre). (cfr annexe 1).

II) Description des champs de la norme RFC-822.

Le précédent chapitre a décrit la structure générale d'un message de RFC-822 : un en-tête divisé en champs et suivi d'un corps. Les briques de base comme les "quoted-strings", "atoms" ... y ont également été définies.

Nous abordons, dans ce paragraphe, l'édifice principal de la norme RFC-822 : la syntaxe des champs. La finesse de la description de leur structure interne est limitée aux "briques adresses", la décomposition de ces dernières en fonction des composants de base faisant l'objet du chapitre IV. L'analyse de la structure générale d'un champ particulier en est ainsi largement facilitée.

L'ordre dans lequel sont présentés les champs n'est pas obligatoire. La seule exigence de ce standard est l'occurrence du corps du message après l'en-tête.

Pour faciliter leur description, la norme RFC-822 les répertorie en classes, elles-même décomposées en sous-classes. Chaque champ est alors syntaxiquement défini avec le service qu'il permet.

Remarquons que la syntaxe RFC-822 présente les champs comme des éléments de protocole mais ne définit pas explicitement des éléments de service. Néanmoins, chaque champ de RFC-822, à l'exception de l'information "trace" peut être vu comme l'implémentation d'un élément de service dont la portée doit être dégagée de la sémantique du champ en question.

Les champs de l'en-tête se répartissent en quatre catégories:

fields = dates

source

1 * destination

*** optional-field**

a) dates.

Cette catégorie est composée d'un champ obligatoire et d'un champ facultatif.

dates = "date" ":" date-time

["resent-date" ":" date-time]

1) "date".

Ce champ indique la date de création du message. Le paramètre "date-time" contient cette information. Sa structure est spécifiée en annexe.

2) "resent-date".

Dans le cas où un utilisateur demande que, pendant son absence, tout courrier reçu soit détourné vers un autre destinataire, les différents champs relatifs aux destinataires sont répétés, précédés par le mot "resent-".

La sémantique du champ est identique qu'il ait ou qu'il n'ait pas le string "resent" relié à son nom. Cependant la présence du préfixe suppose qu'il composait l'en-tête du message origine.

Le champ "resent-date" indique donc la date à laquelle le message a été réexpédié.

b) source.

Trois types d'information appartiennent à cette classe.

source = [trace]

originator

[resent]

1) trace.

L'information "trace" est un outil de vérification du parcours emprunté par un message dans le système de traitement des messages. Elle indique également une route de retour à l'expéditeur.

Route-addr, domain, addr-spec sont des spécifications d'adresses. Leur structure est définie au paragraphe 2 du chapitre IV.

```
trace = "return-path" ":" route-addr

        1 * ( "received" ":"

                ["from" domain]

                ["by" domain]

                ["via" atom]

                * ("with" atom)

                ["id" msg-id]

                ["for" addr-spec]

                   ";" date-time     )

msg-id = "<" addr-spec ">"
```

I) "return-path".

Ce champ est ajouté par le système de transport final qui livre le message à son destinataire. Il est destiné à contenir l'information définitive sur l'adresse et la route de retour vers le créateur du message.

II) "received".

Chaque service de transport qui relaie le message ajoute une copie de ce champ. Cette information peut être très utile pour déterminer des problèmes liés au transport.

"from" spécifie le nom de l'hôte expéditeur et "by" celui du destinataire.

L'utilisation du paramètre "via" indique quel réseau physique, tel que Arpanet ou Phonenet, véhicule le message. Celle du paramètre "with" renseigne le protocole de niveau courrier ou connexion utilisé, tel que le protocole SMTP ou le protocole X25.

L'identifiant interne assigné au message est noté dans le paramètre "id".

Un hôte peut réinterpréter, par expansion ou par transformation, une spécification d'adresse d'un destinataire reçue d'un hôte expéditeur. L'état d'origine de cette spécification, s'il le désire est enregistrée dans le paramètre "for".

2) originator.

Cette catégorie contient toutes les informations en rapport à l'expédition du message.

"mailbox" et "address" sont des spécifications d'adresses

```
originator = "from" ":" mailbox / ( "sender" ":" mailbox
                                    "from" ":" 1 # mailbox )
             ["reply-to" ":" 1 # address]
```

I) "from" / ("sender" et "from")

"from" contient l'identité de l'auteur du message.

"sender" contient l'identité de l'agent (personne, système ou processus) qui expédie le message. Il est généralement utilisé quand l'expéditeur du message n'en est pas l'auteur. Il permet également de signaler dans un groupe d'auteurs, celui qui se charge de l'expédition.

Le champ "from" est toujours obligatoire. Le processus de création doit donc par défaut le remplir d'une seule adresse indiquant l'agent entrant le message. Dans ce cas, l'information du "sender", si présente, devient redondante avec celle du "from". Elle n'a dès lors plus aucune utilité et son usage est déconseillé. Mais si elle diffère du champ "from", elle doit être présente.

Dans l'adresse du champ "sender", une séquence doit correspondre à un agent spécifique (c-à-d un utilisateur humain ou un programme) plutôt qu'une adresse standard. De cette façon le champ identifie un seul agent et non simplement le nom de la boîte-aux-lettres à partir de laquelle le courrier est envoyé. La spécification d'une telle adresse a été détaillée au chapitre précédent.

II) "reply-to".

Ce champ indique à quelle(s) adresse(s) répondre au courrier.

Ce mécanisme est utile dans trois cas typiques. Un auteur qui n'a pas d'adresse machine régulière peut par ce champ spécifier une autre boîte-aux-lettres pour recevoir les réponses. De la même façon, il peut également transmettre les réponses à ce courrier à diverses personnes en insérant leur nom dans ce champ. Enfin comme aide aux téléconférences équipées de services de distribution automatique, l'inclusion de l'adresse de ce service dans le champ "reply-to" de tous les messages soumis à la téléconférence, permet aux participants de garantir la distribution correcte de leur réponses soumises à la conférence.

3) resent.

Les champs de cette catégorie ont la même sémantique que ceux de la catégorie originator avec l'information supplémentaire qu'ils font partie d'un message réexpédié

```
resent = "resent-from" ":" mailbox
                       / ( "resent-sender" ":" mailbox
                           "resent-from" ":" 1 # mailbox )
         ["resent-reply-to" ":" 1 # address ]
```

c) destination.

Dans cette classe sont rassemblés les renseignements concernant les destinataires du message.

```
destination = "to" ":" 1 # address
            / "resent-to" ":" 1 # address
            / "cc" ":" 1 # address
            / "resent-cc" ":" 1 # address
            / "bcc" ":" # address
            / "resent-bcc" ":" # address
```

1) "to".

Ce champ contient l'identité des destinataires primaires du message.

2) "resent-to".

Ce champ est identique au champ "to",mais le message est réexpédié.

3) "cc".

Ce champ contient l'identité des destinataires secondaires du message.

4) "resent-cc".

Ce champ est identique au champ "cc", mais le message est réexpédié.

5) "bcc".

Ce champ contient l'identité de destinataires supplémentaires du message. Le contenu de ce champ n'est pas inclu dans les copies du message envoyé aux destinataires primaires et secondaires.

Suivant le système, le texte de "bcc" est inclus uniquement dans la copie de l'auteur ou dans toutes celles de la liste de "bcc".

6) "resent-bcc".

Ce champ est identique au champ "bcc", mais le message est réexpédié.

d) optional-field.

```
optional-field = "message-id" ":" msg-id
                 / "resent-message-id" ":" msg-id
                 / "in-reply-to" ":" * (phrase / msg-id)
                 / "references" ":" * (phrase / msg-id)
                 / "keywords" ":" # phrase
                 / "subject" ":" * text
                 / "comments" ":" *text
                 / "encrypted" ":" 1 # 2 word
                 / extension-field
                 / user-defined-field
```

1) "message-id".

Ce champ contient un identifiant unique qui refère cette version de ce message. L'unicité de l'identifiant du message est garantie par l'hôte qui le génère. Cet identifiant est destiné à être lu par une machine et n'est pas nécessairement significatif à des humains. Un identifiant de message appartient à exactement une instanciation d'un message particulier. Des révisions de ce message devraient chacune recevoir un nouvel identifiant.

2) "resent-message-id".

Ce champ est identique au champ "message-id", mais le message est réexpédié.

3) "in-reply-to".

Le contenu de ce champ identifie une correspondance antérieure à laquelle ce message répond.

4) "reference.

Le contenu de ce champ identifie d'autres correspondances auxquels ce message se référencie.

5) "keywords".

Ce champ contient des mots clefs ou phrases, séparés par des virgules.

6) "subject".

Ce champ est destiné à fournir un résumé ou indiquer la nature d'un message.

7) "comment".

Ce champ permet d'ajouter du commentaire sur le message sans déranger le contenu du corps du message.

8) "encrypted".

Ce champ est utilisé pour signaler la présence d'un cryptogramme et donner la nature de l'encodage. Le premier paramètre <word> indique le software employé pour coder le corps et le second, optionnel, est destiné à aider le destinataire à choisir la bonne clé de décodage.

9) extension-field.

La classe extension-field regroupe les champs qui ne sont pas définis dans la norme RFC-822, mais dans des documents publiés comme une extension formelle à cette spécification.

Les noms des champs de cette catégorie ne peuvent pas commencer par le string "X-".

10) user-defined-field.

La classe user-defined-field regroupe les champs qui ne sont ni définis dans la norme RFC-822, ni dans des documents publiés comme une extension formelle à cette spécification.

Les noms des champs de cette catégorie devraient commencer par le string "X-".

Syntaxe et sémantique spécifiées, la figure IV.2 reprend l'ensemble des champs qui peuvent apparaître dans tout en-tête d'un message de RFC-822.

Le minimum exigé se limite aux six combinaisons suivantes :

"Dates", "from", et un des six champs : "To", "Resent-To", "cc", "Resent-cc", "bcc", "resent-bcc"

Exemple :

```
Date: 26 Aug 76 1430 EDT

From: George Jones <Group@Host>

Sender: Secy@HOST

To: "Al Neuman"@Mad-Host,
    Sam.Irving@Other-Host

Message-ID: <some.string@HOST>
```

```
"date" ":" date-time
["resent-date" ":" date-time]
"return-path" ":" route-addr
[  1 * ( "received" ":"
           ["from" domain]
           ["by" domain]
           ["via" atom]
           * ("with" atom)
           ["id" msg-id]
           ["for" addr-spec]
           ";" date-time     )   ]
"from" ":" mailbox / ( "sender" ":" mailbox
                                    "from" ":" 1 # mailbox )
["reply-to" ":" 1 # address]
[   "resent-from" ":" mailbox
        / ( "resent-sender" ":" mailbox
            "resent-from" ":" 1 # mailbox )
   ["resent-reply-to" ":" 1 # address]      ]
1*(  "to" ":" 1 # address
     / "resent-to" ":" 1 # address
     / "cc" ":" 1 # address
     / "resent-cc" ":" 1 # address
     / "bcc" ":" # address
     / "resent-bcc" ":" # address  )
*(  "message-id" ":" msg-id
     / "resent-message-id" ":" msg-id
     / "in-reply-to" ":" * (phrase / msg-id)
     / "references" ":" * (phrase / msg-id)
     / "keywords" ":" # phrase
     / "subject" ":" * text
     / "comments" ":" *text
     / "encrypted" ":" 1 # 2 word
     / extension-field
     / user-defined-field )
```

Figure IV.2 : syntaxe générale d'un en-tête d'un message RFC-822

III, Correspondance des services

La syntaxe RFC-822 et les Recommandations X400 spécifient les services offerts aux utilisateurs. Les champs d'un RFC-822 message sont présentés comme élément de protocole mais leur fonctionnalité n'est pas décrite sous forme d'élément de services. Les normes X400, quant à elles, spécifient les éléments de services offerts aux utilisateurs et précisent les éléments de protocole. Nous les avons présentés de manière à ce que nous puissions comprendre leur intérêt par rapport aux éléments de service.

Le but d'une passerelle est de fournir une correspondance entre les éléments de protocole P1 ou P2 et les champs d'un RFC-822 message. Un service demandé par un expéditeur pour un destinataire au-delà de la passerelle, est offert si celle-ci peut établir une correspondance sémantique entre un élément de service X400 et la portée d'un champ RFC-822 et si elle peut échanger les informations sans perte.

Certains éléments de protocole correspondent parfaitement. D'autres ne correspondent en aucune manière et les éléments de services qu'ils représentent ne sont donc pas fournis. Les services restants passent la passerelle dans un but informationnel pour l'utilisateur destinataire : ils sont copiés dans le message.

Le modèle X400 définit deux couches de services : les services offerts à l'utilisateur et les services de transfert de message. La syntaxe RFC-822 ne précise que des champs offrant une sémantique de services à l'utilisateur mais nous supposons qu'il existe un service sous-jacent, appelé le 822-P1 service qui est basé sur trois fonctions :

- identification d'une liste de destinataires (822-P1 Recipients),

- identification d'un adresse de retour d'erreur (822-P1 Originator),

- transfert de messages.

Dans le sens RFC-822 --> X400, la passerelle reçoit un RFC-822 message et les informations du 822-P1 service à partir desquels elle construit un UMPDU, y affecte les valeurs adéquates et transfèrent ce message au monde X400. De même, dans le sens X400 --> RFC-822, elle crée un RFC-822 message et affecte les éléments de protocole dans les champs et les informations du 822-P1 service qui se chargera du transfert dans le monde RFC-822.

Le transfert des informations d'un monde à l'autre doit être le plus complet possible même si a priori, elles n'auront que peu d'intérêt de l'autre côté de la passerelle; en effet, il faut tenir compte du fait que le message transféré par la passerelle a peut-être une destination dans son monde d'origine, ce qui nécessite un nouveau passage de la frontière entre les deux mondes. Il y a donc intérêt de définir des champs d' extension dans le RFC-822 message (cfr annexe) :

- "P1-Message-Id :",
- "X400-Trace :",
- "Original-Encoded-Information-Types :",
- "P1-Content-Type :",
- "Priority :",
- "P1-Recipient :",
- "Deferred-Delivery :",
- "Bilateral-Info :",
- "Obsoletes :",
- "Expiry-Date ",
- "Reply-By :",
- "Importance :",
- "Sensitivity :",
- "Autoforwarded :".

L'utilisateur souhaitant envoyer un message qui nécessite un transfert par une passerelle, doit donner dans la syntaxe de son monde des informations concernant l'autre monde. L'adresse d'un destinataire du monde X400 doit donc être précisé par un expéditeur d'un RFC-822 message selon un protocole, présenté dans le document RFC-987; de même dans le sens inverse (cfr chapitre IV). Les deux algorithmes généraux, définis au chapitre IV, seront utilisés : le premier convertit une adresse de format RFC-822 en ORName, si elle est valide; le second vérifie l'exactitude d'un ORName et le convertit en format RFC-822.

De plus, dans le monde RFC-822, un utilisateur avisé peut utiliser indirectement des services X400 en spécifiant les informations nécessaires dans les champs en extension définis dans le document RFC-987. Evidemment, ces champs n'auront aucun impact dans le cadre strict de RFC-822.

Une politique de traitement des rapports de livraison ou de non-livraison par la passerelle doit être choisie : nous avons estimé que :

- dans le sens RFC-822 --> X400, la passerelle demandera automatiquement un rapport de non livraison avec retour de contenu

- dans le sens X400 --> RFC-822, c'est la passerelle, qui étant sur le système X400, renverra le rapport de livraison ou de non-livraison vers l'expéditeur pour lui signaler que son message est parvenu jusqu'à elle.

a) Interconnexion des services dans le sens RFC-822 -> X400

La passerelle d'un monde RFC-822 vers un monde X400 reçoit en entrée un RFC-822 message et les informations du service 822-P1 sous-jacent. Les arguments du service 822-P1 (822-P1 Originator et 822-P1 Recipients) sont toujours l'adresse de l'expéditeur et l'adresse du(es) destinataire(s) en format RFC-822. Les champs de l'en-tête minimal du RFC-822 message ("Date :", "To :", " From :") sont toujours garnis également tandis que les autres champs existent ou non suivant l'intérêt de l'utilisateur pour les services offerts. De plus, si le RFC-822 message est d'origine X400 ou si l'utilisateur dans le monde RFC-822 veut utiliser indirectement des services X400, il peut également exister des champs en extension. Dans cette direction, le rôle de la passerelle est de construire un UMPDU et y affecter les valeurs adéquates :

- le 822-P1 Originator est à placer dans P1.UMPDUenveloppe.Originator, le(s) 822-P1 Recipient(s) dans P1.UMPDUenveloppe.RecipientInfo et les champs de l'en-tête dans les éléments du P1.UMPDUenveloppe et de P2.Heading,

- le IPMessage sera composé d'un ou deux BodyPart qui seront de type P2.IA5Text. La dernière partie de corps reprendra le corps du RFC-822 message et l'existence du premier sera justifié par la présence de champs d'en-tête RFC-822 non encore en correspondance avec des éléments de protocole X400. Dans ce cas, la première ligne de cette partie de corps sera

" RFC-822 headers : ",

suivie du nom et du corps des champs à transférer.

Voici présentée la construction des éléments de protocole de UMPDU à partir des champs du RFC-822 message :

UMPDU enveloppe :	Correspondance avec les éléments de RFC-822 :
	La valeur à affecter dans les éléments de protocole P1 est générée par :
MPDUidentifier	- soit le champ d'extension "P1-Message-Id:" - soit le champ "Message-Id:" - soit la passerelle elle-même. (cfr remarque 1)
OriginatorORname	- le 822-P1 Originator converti en O/Rname par l'algorithme général
OriginalEncoded-InformationTypes	- le champ en extension "Original-Encoded-Info-Types:"
Contenttype	- "P2"; si la valeur du champ en extension "P1-Content-Type:" n'est pas "P2", le message doit être rejeté.
UAcontentid	- le champ en extension "UA-Content-Id:"
Priority	- le champ en extension "Priority:" si la valeur de ce champ est "escape", le P1-Priority est "normal".

Permessageflag :	(cfr remarque 5)
DiscloseRecipients	- mis à 1 si le champ en extension "P1-Recipient:" existe
ConversionProhibited	
AlternateRecipient-Allowed	- mis à 1
ContentReturnRequest	
DeferredDelivery	- le champ d'extension "DeferredDelivery:"
PerDomainBilateralInfo	
RecipientInfo : (pour tout destinataire)	(cfr remarque 4)
RecipientORname	- soit le(s) 822-P1 Recipient(s) converti(s) en O/Rname par l'algorithme général - soit la passerelle elle-même
ExtensionIdentifier	
PerRecipientFlag :	
ResponsabilityFlag	- mis à 1
ReportRequest	
UserReportRequest	- mis à 01
ExplicitConversion	- pas nécessaire

TraceInformation : (pour chaque domaine) GlobalDomain-Identifier DomainSuppliedInfo : Arrival Defined Action Converted- InformationTypes PreviousGlobal- DomainIdent	(cfr remarque 6)

Composantes de l'en-tête d'un IM-UAPDU	Correspondance avec les champs de RFC-822 :
	La valeur à affecter dans les éléments de protocole P2 est générée par :
IPMessageId	- soit le champ "Message-Id:" - soit la passerelle elle-même (cfr remarque 1)
Originator	- soit le champ "Sender:" converti en O/Rname par l'algorithme général - soit le champ "From:" converti en O/Rname par l'algorithme général (cfr remarque 2)
AuthorizingUsers	- le champ "From:" (converti en O/Rname par l'algorithme général) si le champ "Sender:" existe. (cfr remarque 2)
PrimaryRecipients	- le champ "To:" (cfr remarque 3)
CopyRecipients	- le champ "Cc:" (cfr remarque 3)
BlindCopyRecipients	- le champ "Bcc:" (cfr remarque 3)

InReplyTo	- le champ "InReplyTo:" (cfr remarque 7)
Obsoletes	- le champ en extension "Obsoletes:"
CrossReferences	- le champ "References:"
Subject	- le champ "Subject:"
ExpiryDate	- le champ en extension "Expiry-Date:"
ReplyBy	- le champ en extension "Reply-By:"
ReplyToUsers	- le champ "Reply-To:" (cfr remarque 2)
Importance	- le champ en extension "Importance:" Si la valeur de ce champ est "escape", le niveau d'importance est "normal"
Sensitivity	- le champ en extension "Sensitivity:" Si la valeur de ce champ est "escape", le niveau est "normal"
Autoforwarded	- (cfr remarque 8)

Certains champs du RFC-822 message n'ont pas été transférés dans un élément de protocole P1 ou P2 : "Resent :", "Keywords :", "Comments :", "Encrypted :". N'apportant qu'une information optionnelle, ils sont recopiés intégralement dans la première partie du corps précédés par "RFC-822 Headers". Le champ "Return-path :" n'est pas à prendre en compte car une valeur ne lui est affectée par un destinataire que s'il se trouve dans le monde RFC-822. Le corps du RFC-822 message est transféré dans le corps du IM-UAPDU de type IA5String. Un utilisateur avisé peut utiliser indirectement des services X400 en spécifiant de manière correcte les champs en extension, ceux-ci n'étant pas fournis dans le cadre RFC-822 :

- "Priority :",
- "Deferred-Delivery :",
- "Obsoletes :",
- "Expiry-Date :",
- "Reply-By :",
- "Importance :",
- "Sensitivity :".

Il nous paraît utile de définir des champs en extension supplémentaires pour offrir aux utilisateurs RFC-822 des services X400 tels que :

- un champ d'extension pour permettre à un utilisateur dans le monde RFC-822 de lancer une sonde dans le monde X400. La réponse à cette sonde atteindra la passerelle qui transférera ce rapport en un RFC-822 message que l'utilisateur pourra interpréter,

- un champ en extension pour exiger à la passerelle une politique bien précise en matière de rapport de livraison ou de non-livraison et de retour de contenu.

Une étude précise de la syntaxe de ces champs en extension et des actions de la passerelle n'a pas été l'objet de ce travail mais pourra augmenter l'intérêt d'une telle passerelle.

b) Interconnexion des services dans le sens X400 -> RFC-822

La passerelle d'un monde X400 vers un monde RFC-822 reçoit en entrée un UMPDU composé de son enveloppe et de son contenu qui est un IM-UAPDU, lui-même contenant les en-têtes et le corps du message. En puisant les valeurs dans les éléments de protocole corrects, la passerelle créera et garnira les champs et le corps d'un RFC-822 message, de même que les arguments du service 822-P1. Les cas des Service MPDU (sonde et rapport de livraison ou de non-livraison) sont traités dans le paragraphe suivant.

Voici l'affectation adéquate des champs du RFC-822 message et des arguments du service 822-P1 :

Eléments 822-P1 et RFC-822 message :	Correspondance avec les éléments de protocole P1 et P2 du message X400 :
Eléments 822-P1 :	La valeur à affecter dans les éléments 822-P1 est générée par :
822-P1 Originator	- P1.UMPDU.enveloppe-Originator converti en format RFC-822 par l'algorithme général
822-P1 Recipients	- P1.RecipientInfo (cfr remarque 4)
En-tête du RFC-822 message :	La valeur à affecter dans les champs de l'en-tête du RFC-822 message est générée par :
Date :	- DomainSuppliedInfo.arrival de la première composante de P1.TraceInformation
From :	- soit P2.Heading-AuthorizingUsers converti en format RFC-822 par l'algorithme général - soit P2.Heading.Originator converti en format RFC-822 par l'algorithme général - soit P1.UMPDU.enveloppe-Originator converti en format RFC-822 par l'algorithme général (cfr remarque 2)

Sender :	- P2.Heading.Originator si l'élément de protocole P2.Heading.AuthorizingUser existe, converti en format RFC-822 par l'algorithme général (cfr remarque 2)
To :	- P2.Heading-PrimaryRecipients (cfr remarque 3)
Cc :	- P2.Heading.CopyRecipients (cfr remarque 3)
Bcc :	- P2.Heading-BlindCopyRecipients (cfr remarque 3)
Reply-To :	- P2.Heading.ReplyToUsers (cfr remarque 2)
Message-Id :	- P2.Heading.IPMessageId (cfr remarque 1)
In-Reply-To :	- P2.Heading.InReplyTo (cfr remarque 7)
References :	- P2.Heading.CrossReferences

Keywords :	
Subject :	- P2.Heading.Subject converti par l'algorithme de conversion de caractère Printablestring en caractère ASCII
Comments :	
Encrypted :	

Les champs d'un RFC-822 étant plus restreints, beaucoup d'éléments de protocole de P1 ou de P2 n'ont pas d'équivalents dans le monde RFC-822. Dans le cas où le RFC-822 message construit à partir d'un UMPDU doit atteindre à nouveau le monde X400, il est important de garder ces informations dans le message. C'est une des raisons pour lesquelles les champs en extension ont été définis précédemment et voici leur affectation :

Champs en extension :	Correspondance avec les éléments de protocole : La valeur à affecter dans les éléments de RFC-822 est généré par :
"P1-Message-Id :"	- P1.UMPDUenveloppe-MPDUIdentifier
"Original-Encoded-Information-Types :"	- P1.UMPDUenveloppe-OriginalEncodedInfoTypes
"UA-Content-Id :"	- P1.UMPDUenveloppe-UAContentType
"P1-Priority :"	- P1.UMPDUenveloppe-Priority
"P1-Recipient :"	- ce champ est généré si P1.UMPDUenveloppe-PerMessageFlag-DiscloseRecipient est mis à 1 et contient tous les P1.UMPDUenveloppe.RecipientInfo
"Deferred-Delivery :"	- P1.UMPDUenveloppe-DeferredDelivery
"Bilateral-Info :"	- P1.UMPDUenveloppe-PerDomainBilateralInfo
"X400-Trace :"	- P1.UMPDUenveloppe-traceInformation

"Expiry-Date :"	- P2.Heading.ExpiryDate
"Reply-By :"	- P2.Heading.ReplyBy
"Importance :"	- P2.Heading.Importance
"Sensitivity :"	- P2.Heading.Sensitivity
"Autoforwarded :"	- (cfr remarque 8)

Les champs d'un RFC-822 étant plus restreints, beaucoup d'éléments de protocole de P1 ou de P2 n'ont pas d'équivalents dans le monde RFC-822. Dans le cas où le RFC-822 message construit à partir d'un UMPDU doit atteindre à nouveau le monde X400, il est important de garder ces informations dans le message. C'est une des raisons pour lesquelles les champs en extension ont été définis précédemment et voici leur affectation :

Le P2.Body est transféré dans le corps de RFC-822 message après la conversion de chaque P2.BodyPart en ASCII. si le P2.Body contient un P2.BodyPart non cité ci-dessous, le message entier est rejeté. S'il y a exactement deux P2.BodyPart et si le pemier débute avec le texte "RFC-822 Headers :", alors le reste de cette partie de corps sera traité comme des informations supplémentaires pour le RFC-822 message. Dans le cas de P2.BodyPart multiples, la correspondance se fait de manière individuelle :

- P2.IA5Text : la correspondance est immédiate,

- P2.TTX : si un P1.Teletex.NonBasicParams est mis à 1, le message est rejeté. Sinon, il est converti en ASCII comme cela est présenté dans le chapitre IV,

- P2.SFD : un SFD sera converti en ASCII quand il sera placé en 79 colonnes,

- P2.ForwardedMessage : le P2.ForwardedMessage.Delivery et le P2.ForwardedMessage.DeliveryInformation ne sont pas pris en compte. Le IM-UAPDU sera transféré suivant les spécifications de la passerelle.

Il est évident que la qualité des services X400 est toute différente quand le destinataire est un utilisateur du monde RFC-822. Un site RFC-822 n'est pas conçu pour répondre à une sonde et à une demande de rapport de livraison ou de non-livraison. Dès lors, c'est la passerelle, étant encore en monde X400 qui se charge d'offrir ces services, détériorés car la sonde ne précise son efficacité que jusque la passerelle et les rapports de livraison ne concernent que l'accès au monde RFC-822.

D'autres services n'ont plus aucun intérêt à partir du monde RFC-822 :

- choix du degré de livraison,
- permission de livrer à un destinataire redirigé,
- livraison différée (la passerelle pouvant se charger de retarder le passage en monde RFC-822),
- indication de transfert automatique vers un autre destinataire,
- indication de la date d'expiration.

c) Remarques.

Remarque 1 : Correspondance entre les identificateurs du message X400 et du RFC-822 message.

La description des correspondances entre les identificateurs du message X400 (P1.UMPDUenveloppe.MPDUIdentifier et P2.Heading.IPMessageId) et les champs "Message-Id :" et "P1-Message-Id :" demande une présentation précise des normes X400 et RFC-822 sur ces éléments. La syntaxe RFC-822 définit le champ optionnel "Message-Id :" comme étant "<"local-part@domain">". Un champ en extension "P1-Message-Id :" peut également exister dans un RFC-822 message et sa syntaxe a été définie précédemment dans ce paragraphe. Un message X400 comprend deux identificateurs obligatoires P1.UMPDUenveloppe.MPDUIdentifier et P2.Heading.IPMessageId (cfr annexe)

La correspondance RFC-822 --> X400 s'établit de la manière suivante :

- le P1.UMPDUenveloppe.MPDUIdentifier contient le champ en extension "P1-Message-Id :", le champ "Message-Id:" si le premier n'existe pas ou est généré par la passerelle si les deux précédents n'existent pas. Le champ "Message-Id :" est transféré comme suit :

P1.CountryName = " "
P1.ADmnistrationName = domain
P1.IA5String = local-part

- P2.Heading.IPMessageId contient le champ "Message-Id :" s'il existe ou est généré par la passerelle. Le "Message-Id :" est transféré comme suit : la local-part est analysée comme

[printablestring "*"] local-part réelle.

Si la partie printablestring existe, elle est transférée dans P2.Heading.IPMessageId.PrintableString. Sinon, ce dernier est "RFC-822". Si la local-part réelle existe, l'identificateur sans sa partie 'printablestring "*" ' est converti en ORName par l'algorithme général présenté dans le chapitre IV.

La correspondance X400 --> RFC-822 est la suivante :

Le champ "Message-Id :" reçoit le P2.Heading.IPMessageId : le ORName est converti en format RFC-822 par l'algorithme général et le PrintableString est converti en ASCII; si ce dernier est différent de "RFC-822", il est ajouté devant la local-part avec "*" comme séparateur. Il est à noter que CRLF est converti en SPACE, ce qui rend la correspondance irréversible dans certains cas. Le champ en extension "P1-Message-Id :" est créé pour contenir le P1.UMPDUenveloppe.MPDUIdentifier.

Remarque 2 : Correspondance entre un P2.ORDescriptor et une adresse du monde RFC-822.

La correspondance entre les champs "From :","Sender :", "To :", "Cc :", "Bcc :", "Reply-To :" et les éléments P2.Heading.AuthorizingUsers, -.PrimaryRecipients, -.CopyRecipients, -.BlindCopyRecipients, -.ReplyToUsers nécessite la conversion d'une adresse de format RFC-822 en un P2.ORDescriptor et vice-versa. Le P2.ORDescriptor est composé d'un ORName et de deux strings (FreeFormname en T61 et Telephonenumber en PrintableString). La syntaxe RFC-822 spécifie qu'une adresse peut être un "mailbox" ou un groupe de "mailbox" précédé d'une phrase, un "mailbox" pouvant également comprendre une phrase (cfr chapitre IV.1). Une adresse de format RFC-822 est convertie en P2.ORDescriptor de la manière suivante :

- "addr-spec" ou "route-addr" est converti en P2.ORDescriptor.ORName par l'algorithme général présenté précédemment.

- Un string est construit à partir de la phrase si elle existe ou de commentaires, s'il y en a.

- Ce string est ensuite assigné dans P2.ORDescriptor.FreeFormname et le P2.ORDescriptor.Telephonenumber est omis.

Le P2.ORDescriptor est transformé en adresse de format RFC-822 par les traitements qui suivent :

- le P2.ORDescriptor.ORName est converti en format RFC-822 par l'algorithme général,

- si P2.ORDescriptor.FreeFormname est présent, il est converti en ASCII et est placé comme phrase devant "route-addr". Si P2.ORDescriptor.Telephonenumber existe, il est placé en commentaire.

Remarque 3 : Correspondance entre les champs "To :", "Cc :", "Bcc :" et les éléments P2.Heading.PrimaryRecipients, -.CopyRecipients, -.BlindCopyRecipients

P2.Heading.PrimaryRecipients, -.CopyRecipients, -.BlindCopyRecipients sont composés de trois champs :

- P2.ORDescriptor (converti à partir d'une adresse ou en une adresse de format RFC-822 comme le précise la remarque 2),

- ReportRequest,

- ReplyRequest.

Dans le sens RFC-822 --> X400, les deux derniers éléments ne sont pas considérés. Par contre, dans l'autre, si P2.Recipient.ReportRequest spécifie l'avis de réception, le commentaire "(Receipt Notification Request)" sera ajouté à l'adresse. De même, si P2.Recipient.ReplyRequest existe, les commentaires "(Reply Requested)" ou "(Reply Not Requested)" y seront insérés, suivant la valeur de ce bit.

Remarque 4 : Correspondance entre les éléments 822-P1 Recipients et P1.UMPDUenveloppe.RecipientInfo.

L'élément de protocole P1.UMPDUenveloppe.RecipientInfo est composé d'un ORName, de ExtensionIdentifier, de PerRecipientFlag et de ExplicitConversion; le 822-P1 Recipient est une adresse de format RFC-822. L'algorithme général du chapitre IV convertit l'adresse RFC-822 en ORName et inversément. A chaque 822-P1.Recipient correspond un P1.Recipient.

Dans le sens RFC-822 --> X400, la correspondance s'établit comme suit :

- ExtensionIdentifier est généré par le monde X400
- PerRecipientFlag :
  - ResponsabilityFlag est mis à 1,
  - ReportRequest a la valeur basic,
  - UserReportRequest est affecté à basic,
- ExplicitConversion est ignoré.

Les valeurs des trois sous-champs de PerRecipientFlag sont dues au choix de la politique de traitement de rapport de livraison ou de non-livraison. De plus, si le champ en extension "P1-Recipient :" existe, toutes les adresses qui y sont répertoriées et qui ne correspondent pas aux 822-P1 Recipients sont ajoutées à la liste de P1.UMPDUenveloppe.RecipientInfo.

Dans le sens X400 --> RFC-822, la correspondance est la suivante :

Si ResponsabilityFlag est mis à 1, la valeur des champs de P1.UMPDUenveloppe.RecipientInfo fait agir la passerelle de la sorte :

- ExtensionIdentifier n'est pas utilisé

- PerRecipientFlag : la passerelle enverra un rapport de non-livraison avec retour du contenu vers l'expéditeur si le message ne peut aller au-delà de la passerelle; si le ReportRequest a la valeur Confirmed ou Audit-and-Confirmed, elle enverra un rapport de livraison si le message est confié au monde RFC-822; dans la passerelle, seront spécifiés l'identité de la passerelle, la nature du rapport et le fait que rien n'est certifié sur la livraison du message à son destinataire dans le monde RFC-822.

- ExplicitConversion : si ce champ existe, le ORName spécifiant le destinataire concerné par la conversion est rejeté.

Remarque 5 : L'élément de protocole P1.UMPDUenveloppe.PerMessageFlag.

Dans le sens RFC-822 --> X400, le P1.UMPDUenveloppe.PerMessageFlag est construit de la manière suivante :

- DiscloseRecipient est mis à 1 si le champ en extension "P1-Recipient :" existe puisque ce champ contiendra une valeur si le message vient d'un site X400 et si cet élément de protocole était mis à 1,

- ConversionProhibited n'a pas d'importance,

- AlternateRecipientAllowed est mis à 1, ce qui provoquera l'exécution de cet élément de service dans le monde X400,

- ContentReturnRequest est mis à 1, étant donné la politique présentée précédemment.

Dans le sens X400 --> RFC-822, P1.UMPDUenveloppe.PerMessageFlag influence le RFC-822 message de la manière suivante :

- si DiscloseRecipient est mis à 1, le champ en extension "P1-Recipient :" est généré et contient tous les P1.UMPDUenveloppe.RecipientInfo,

- si ConversionProhibited est mis à 1, le message sera rejeté s'il contient un P2.BodyPart de type différent de P2.IA5Text ou de P2.ForwardedIPMessage,

- AlternateRecipientAllowed & ContentReturnRequest sont ignorés.

Remarque 6 : L'élément de protocole P1.UMPDUenveloppe.TraceInformation.

Dans le sens RFC-822 --> X400, les composantes de TraceInformation sont construites comme suit :

- la première composante est établie à partir du champ "Date :" :

```
GlobalDomainIdentifier :
      CountryName          = " "
      AdministrationName = " "
DomainSuppliedInfo :
      arrival = date du champ "Date :"
      action  = "relayed"
```

- la dernière composante est générée par la passerelle :

```
GlobalDomainIdentifier :
      valeur locale de la passerelle
DomainSuppliedInfo :
      arrival = date courante
      action  = "relayed"
```

- les composantes intermédiaires sont construites à partir du champ "Received :" ou du champ en extension "X400-Trace :", suivant l'ordre chronologique. La correspondance avec "X400-Trace :" est immédiate. Une composante de "Received :" génère une composante de TraceInformation de la manière suivante :

```
GlobalDomainIdentifier :
      CountryName          = " "
      AdministrationName = domaine du champ "by"
                           ou " " s'il n'existe pas
DomainSuppliedInfo :
      arrival = date de "Received :"
      action  = "relayed"
      previous :
            GlobalDomainIdentifier :
                  CountryName          = " "
                  AdministrationName = domaine de
                                       "From"
                  et tous les autres champs
                  "Via", "Id", "For" entre
                  parenthèses, après être
                  converti de caractère
                  ASCII en PrintableString.
```

Dans le sens X400 --> RFC-822, le champ "Date :" reprend la date de DomainSuppliedInfo arrival de la première composante de P1.UMPDUenveloppe.TraceInformation. Les composantes de la TraceInformation sont enregistrées dans le champ en extension "X400-Trace :".

Remarque 7 : Correspondance entre les éléments de protocole P2.Heading.InReplyTo et le champ "In-Reply-To :".

Dans le sens RFC-822 --> X400, la correspondance est la suivante :

si l'adresse du champ "In-Reply-To :" ne comprend pas de phrase et s'il n'existe qu'une seule composante, la valeur du champ est convertie en ORName par l'algorithme général; sinon, tout le champ est transféré dans le P2.BodyPart.

Dans le sens X400 --> RFC-822, le contenu de P2.Heading.InReplyTo est converti en adresse de format RFC-822 par l'algorithme général et enregistré dans le champ "In-Reply-To :".

Remarque 8 : L'élément de protocole P2.Heading.Autoforwarded

Pour un message passant du monde RFC-822 vers un système X400, il peut exister un champ en extension "Autoforwarded :". Si la valeur est "TRUE", alors il y aura un ou plusieurs champs commençant par le string "Autoforwarded-". Ceux-ci seront utilisés, après avoir supprimé "Autoforwarded-", pour construire le IPMessage et P2.Heading.Autoforwarded aura la valeur "TRUE".

Dans le sens X400 --> RFC822, la correspondance dépend de la valeur de P2.Heading.Autoforwarded. S'il est mis à "FALSE", cette même valeur est affectée dans le champ en extension "Autoforwarded :". S'il est mis à "TRUE", si le P2.Body ne consiste pas en un simple P2.ForwardedIPMessage , une erreur est renvoyée et le message supprimé. Sinon, quatre actions seront exécutées par la passerelle :

- la correspondance est établie entre les éléments de protocole de X400 et les champs de RFC-822,

- "AutoForwarded-" est placé devant tous ces champs,

- le champ en extension "AutoForwarded :" est mis à la valeur "TRUE",

- le P2.ForwardedIPMessage est transféré dans le corps du RFC-822 message.

Interconnexion entre systèmes de courrier électronique : spécification d'une passerelle et implémentation de la correspondance des adresses.

Mémoire présenté par

Pascal Degueldre

et

Vincent Fauvarque

en vue de l'obtention du titre de

LICENCIE ET MAITRE EN INFORMATIQUE

Promoteur du mémoire :

Ph. VAN BASTELAER

Année Académique 1987-1988

TOME II

CHAPITRE VI : IMPLEMENTATION.

Suite à la spécification générale d'une passerelle entre les deux mondes RFC-822 et X400, le chapitre présent développe une première étape de son implémentation.

En fonction de la démarche adoptée au cours des précédents chapitres, il se dégage deux phases dans l'élaboration de la passerelle : la création d'un logiciel de conversion d'adresses suivi de la création d'un logiciel de conversion de services, le logiciel d'adresses restant une "boîte noire" pour ce dernier.

Malheureusement, vu le temps imparti et la difficulté de la tâche, seul le logiciel de conversion d'adresses de RFC-822 vers X400 est implémenté dans le cadre de ce mémoire.

Ce dernier est basé sur l'algorithme général de RFC-822 vers X400 développé au chapitre IV. Certaines restrictions y ont été apportées. Ainsi, l'étape de construction de l'ORName selon sa représentation standard est abandonnée pour une représentation plus compréhensible à l'être humain. De même, la syntaxe STD-orname est réduite à un ensemble d'attributs afin d'obtenir un contrôle plus significatif sur leur type :

```
std-orname = 1*("/" attribute "=" value) "/"

attribute = "C" / "A" / "P" / "X121" / "T-ID" / "O" / "OU"
          / "UA-ID" / "S" / "GI" / "I" / "GQ" / "PN"
          / "RFC-822" / "JNT-MAIL" / "UUCP"

value = std-printablestring

std-printablestring = *(std-char / std-pair)

std-pair = "$" (ps-delim / ps-char)

std-char = <ps-delim et ps-char sauf "/" et "=">
```

Puisqu'il s'agit de l'implémentation d'un logiciel, la démarche adoptée dans ce travail est celle prônée par le cours de méthodologie de développement de logiciels enseigné à l'institut d'Informatique des Facultés Notre-Dame de la Paix de Namur.[A. van Lamsweerde]

La méthodologie défendue propose d'identifier les composants du problème et de les organiser en une structure modulaire c.-à-d. les composants qui la forment et les relations qui lient ses composants sont tels que les attributs de tout module peuvent être définis de manière simple et précise et tout composant jouit des qualités de forte capacité cachée d'information, de forte cohésion et de faible degré de couplage.[]

Il s'agit de créer une architecture logique c.-à-d. indépendante de la machine et du langage utilisé.

Diverses relations entre les composants peuvent être envisagées, nous retenons la relation "utilise" qui offre des garanties quant à la robustesse, la fiabilité et la maintenance du logiciel.

La définition d'une relation utilise est :

"A utilise B" ssi la validité du composant A dépend de la disponibilité d'une version correcte du composant B.

Parmi les méthodes de spécification de modules, la méthode par assertion offre rigueur et concision et réduit les défauts tels que le bruit, le silence... Chaque module est décrit par la liste de ses arguments et de ses résultats et une paire d'assertions, précondition et postcondition.

Une précondition est une condition qui doit expliciter les propriétés des arguments. Ces propriétés doivent être satisfaites avant toute exécution du module pour que ce module s'exécute correctement.

Une postcondition est une condition qui doit expliciter les propriétés des résultats du module. Ces propriétés doivent être satisfaites en fin d'exécution du module si ce module s'est exécuté correctement.

Les différents modules découlant de l'analyse du problème et le programme principal seront spécifiés à l'aide de cette méthode.

L'étape suivant l'architecture logique dans le développement d'un logiciel est l'architecture physique. Elle tient compte des caractéristiques propres à la machine et au langage utilisé.

Il reste alors à coder et tester les algorithmes.

Dans un premier temps, nous donnons les spécifications du programme principal lié à l'algorithme général. Nous présentons alors les divers modules dérivés de la structuration du problème et une représentation graphique des liens entre les composants.

Une autre découpe du système peut se baser sur un diagramme de flux où chaque module correspond à une seule phase de traitement avec toute l'information véhiculée à travers tous les modules. Mais cette stratégie engendre des interfaces plus complexes et des difficultés de modifiabilité.

Une spécification de chaque module est alors explicitée. L'architecture physique est limitée à l'environnement dans lequel notre programme est intégré; la codification des algorithmes est présentée en annexe. Le chapitre se termine par les tests et résultats obtenus.

I) Spécification du programme.

Arguments : adresse ∈ STRING,
822-P1 ∈ ENTIER,
attribut ∈ STRING,
quoted ∈ ENTIER.

Préconditions :
822-P1 = 1 si l'adresse est un 822-P1 Recipient,
= 0 sinon;
quoted = 1 si le réseau source ne supporte pas la syntaxe quoted-string,
= 0 sinon;
attribut = "RFC-822=", "JNT-MAIL=", "UUCP=".

Résultats : X400 ∈ STRING.

Postconditions :
X400 est sous la forme :
attribute = value *(";" *(SPACE) attribute = value)
et X400 est une forme de ORName correcte
ou
X400 est indéterminé et une liste d'erreurs est transmise.

II) Découpe en modules.

L'analyse du programme a engendré l'architecture logique composée de cinq modules reliés par la relation utilise selon le schéma de la figure VI.1. :

Module STRING : un élément de type STRING est une chaîne de caractères terminée par un symbôle de fin de chaîne. Ce module regroupe un ensemble d'opérations exécutables sur un élément de type STRING .

Module TABLE : une table est une structure à deux composantes : une entrée et une sortie qui sont des éléments de type STRING. A chaque entrée doit correspondre une sortie et toutes les entrées doivent être distinctes. Ce module regroupe un ensemble d'opérations exécutables sur les tables comme la lecture des entrées et des sorties.

Module ANALYSE : ce module regroupe un ensemble de primitives d'analyse syntaxique d'une adresse ou d'une partie d'adresse.

Module MAPPING : Ce module regroupe toutes les correspondances de l'algorithme général.

Module ERREUR : Ce module gère les erreurs du programme

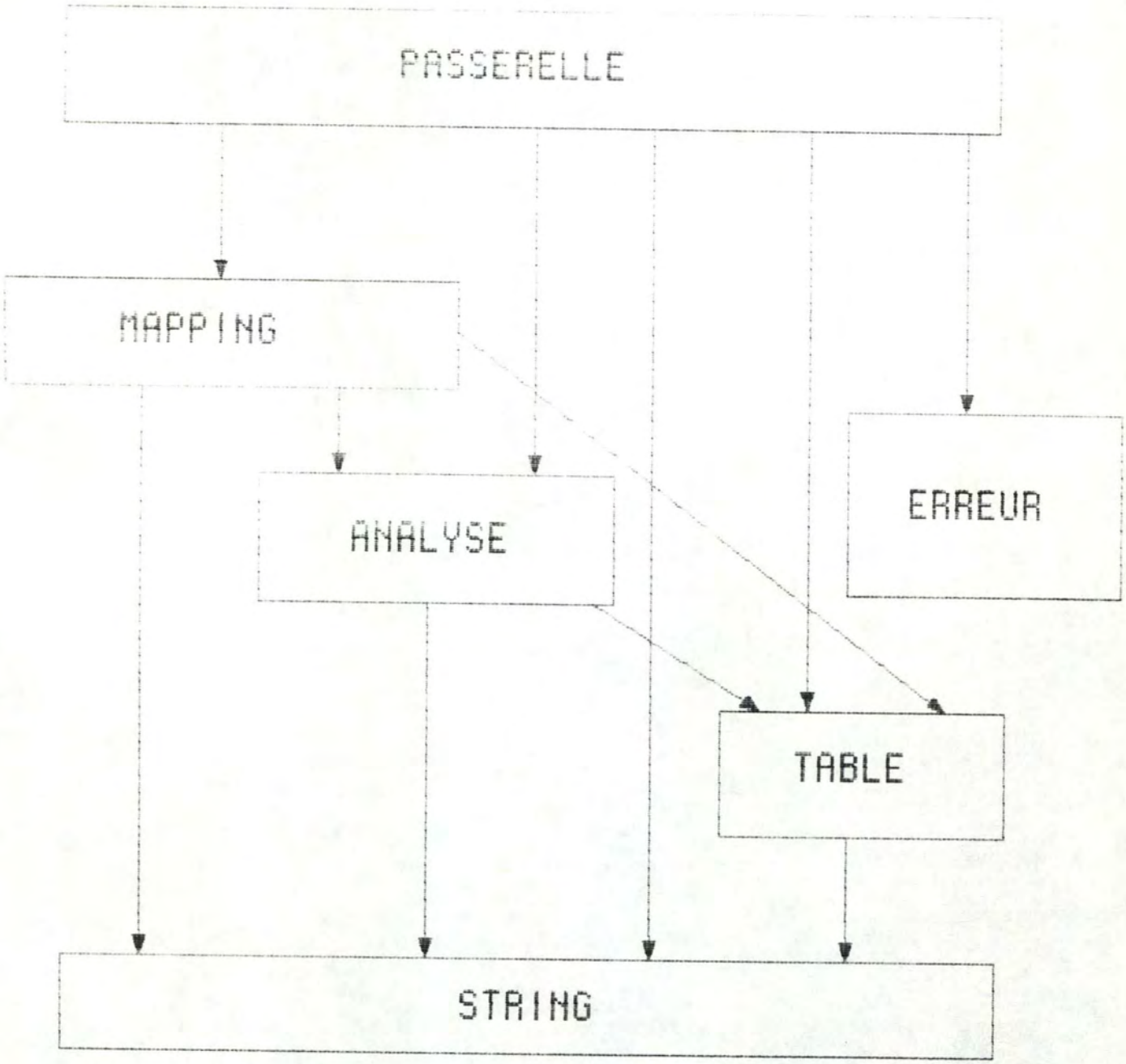


FIGURE VI.1. : ARCHITECTURE LOGIQUE DU PROGRAMME.

III) Spécification des modules.

a) Module string.

Primitive1 : creer.

But : créer un élément de type STRING ou s'il existe le réinitialiser.

Arguments : / .

Préconditions : / .

Résultats : string ϵ STRING.

Postconditions : longueur(string) = 0.

Primitive2 : longueur.

But : obtenir le nombre de caractères constituant un élément de type STRING.

Arguments : string ϵ STRING.

Préconditions : / .

Résultats : long ϵ ENTIER.

Postconditions : long ≥ 0 et est le nombre de caractères de string.

Primitive3 : caractere.

But : obtenir le caractère apparaissant en une position demandée dans un élément de type STRING.

Arguments : string ∈ STRING,
position ∈ ENTIER.

Préconditions : 1 ≤ position ≤ longueur(string).

Résultats : car ∈ CHAR.

Postconditions : car est le caractère de string situé en position.

Primitive4 : ajout.

But : ajouter un caractère quelconque à la fin d'un élément de type STRING.

Arguments : string1 ∈ STRING,
car ∈ CHAR,
long ∈ ENTIER.

Préconditions : long = longueur(string1) ≥ 0.

Résultats : string2 ∈ STRING.

Postconditions : longueur(string2) = long+1
et
pour tout i tel que 1 ≤ i ≤ long on a :
caractere(string1,i) = caractere(string2,i)
et
caractere(string2,long+1) = car.

Primitive5 : cherchercar.

But : obtenir la première occurrence d'un caractère après une position définie dans un élément de type STRING.

Arguments : string ∈ STRING,
pos1 ∈ ENTIER,
car ∈ CHAR.

Préconditions : 0 ≤ pos1 < long,
long = longueur(string).

Résultats : pos2 ∈ ENTIER.

Postconditions :
pos1 < pos2 ≤ long
et
caractere(string,pos2) = car
et
pour tout i tel que pos1 < i < pos2 on a :
caractere(string,i) <> car
ou
pos2 = 0.

Primitive6 : derniercar.

But : obtenir la dernière occurrence d'un caractère dans un élement de type STRING.

Arguments : string ∈ STRING,
car ∈ CHAR,
long ∈ ENTIER.

Préconditions : long = longueur(string) > 0.

Résultats : pos ∈ ENTIER.

Postconditions :
0 < pos ≤ long
et
caractere(string,pos) = car
et
pour tout i tel que pos+1 ≤ i ≤ long on a :
caractere(string,i) <> car
ou
pos2 = 0.

Primitive7 : nbrecar.

But : obtenir le nombre d'occurrences d'un caractère dans une partie d'un élément de type STRING.

Arguments : string1 ∈ STRING,
car ∈ CHAR,
pos1 ∈ ENTIER,
pos2 ∈ ENTIER.

Préconditions : 1 ≤ pos1 ≤ pos2 ≤ longueur(string1).

Résultats : nbre ∈ ENTIER.

Postconditions :
string1 est inchangé
et
0 ≤ nbre ≤ pos2-pos1+1
et
#(i tq pos1 ≤ i ≤ pos2 et caractere(string1,i) = car) = nbre.

Primitive8 : prendre.

But : obtenir un élément de type STRING, sous-élément d'un autre élément de type STRING.

Arguments : string1 ∈ STRING,
pos1 ∈ ENTIER,
pos2 ∈ ENTIER.

Préconditions : 1 ≤ pos1 ≤ pos2 ≤ longueur(string1).

Résultats : string2 ∈ STRING.

Postconditions :
string1 est inchangé
et
longueur(string2) = pos2-pos1+1
et
pour tout i tel que 1 ≤ i ≤ longueur(string2) on a :
caractere(string2,i) = caractere(string1, pos1-1+i).

Primitive9 : concat.

But : obtenir un élément de type STRING auquel on a concaténé un autre élément de type STRING.

Arguments : string1 ∈ STRING,
string2 ∈ STRING.

Préconditions : / .

Résultats : string1 ∈ STRING.

Postconditions :
pour tout i tel que 1 ≤ i ≤ longueur(string1) on a :
caractere(string1,i) inchangé
et
pour tout i tel que 1 ≤ i ≤ longueur(string2) on a :
caractere(string1,longueur(string1)+i) = caractere(string2,i)
et
longueur(string1) = longueur(string1) + longueur(string2).

Primitive10 ; copier.

But : obtenir un élément de type STRING copie d'un autre élément de type STRING.

Arguments : string1 ∈ STRING.

Préconditions : / .

Résultats : string2 ∈ STRING.

Postconditions :
string1 est inchangé
et
pour tout i tel que 1 ≤ i ≤ longueur(string1) on a :
caractere(string2,i) = caractere(string1,i)
et
longueur(string1) = longueur(string2).

Primitive11 ; replace.

But : obtenir un élément de type STRING dans lequel on a remplacé toute occurrence d'un symbole par un autre symbole.

Arguments : string ∈ STRING,
car1 ∈ CHAR,
car2 ∈ CHAR.

Préconditions : longueur(string) ≥ 0.

Résultats : string ∈ STRING.

Postconditions :
pour tout i 1 ≤i ≤ longueur(string) tq
caractere(string,i) = car1 on a :
caractere(string,i) = car2
et
pour tout i 1 ≤ i ≤ longueur(string) tq
caractere(string,i) <> car1 on a :
caractere(string,i) inchangé.

Primitive12 : comparer.

But : vérifier si deux éléments de type STRING sont identiques.

Arguments : string1 ∈ STRING,
string2 ∈ STRING.

Préconditions : / .

Résultats : egal ∈ BOOLEAN.

Postconditions :
egal = vrai
ssi
pour tout i tel que 1 ≤ i ≤ longueur(string1) on a :
caractere(string1,i) = caractere(string2,i)
et
longueur(string1) = longueur(string2).

Primitive13 : inserer.

But : obtenir un élément de type STRING dans lequel on a inséré un autre élément de type STRING en une position définie.

Arguments : string1 ∈ STRING,
string2 ∈ STRING,
pos ∈ ENTIER.

Préconditions : 1 ≤ pos ≤ longueur(string1)+1.

Résultats : string1 ∈ STRING.

Postconditions :
pour tout i tel que 1 ≤ i ≤ pos-1 on a :
caractere(string1,i) est inchangé
et
pour tout i tel que pos ≤ i ≤ longueur(string1) on a :
caractere(string1,longueur(string2)+i) = caractere(string1,i)
et
pour tout i tel que 1 ≤ i ≤ longueur(string2) on a :
caractere(string1,pos-1+i) = caractere(string2,i)
et
longueur(string1) = longueur(string1) + longueur(string2).

Primitive14 : changer.

But : remplacer dans un élément de type STRING un caractère en une position déterminée par un autre caractère.

Arguments : string ∈ STRING,
pos ∈ ENTIER,
car ∈ CHAR.

Préconditions : 1 ≤ pos ≤ longueur(string).

Résultats : string ∈ STRING.

Postconditions :
caractere(string,pos) = car
et
pour tout i tel que 1 ≤ i ≤ longueur(string) et i <> pos
on a : caractere(string1,i) est inchangé

b) Module table.

Primitive1 : liretable.

But : lire successivement toutes les entrées d'une table et les sorties correspondantes.

Arguments : table ∈ STRING.

Préconditions : table existe.

Résultats : element ∈ STRING.

Postconditions : element contient successivement l'entrée de table et sa sortie.

Primitive2 : chercher.

But : vérifier qu'un élément de type STRING correspond à une entrée de la table et en déduire la sortie.

Arguments : table ∈ STRING,
string ∈ STRING.

Préconditions : table existe.

Résultats : sortie ∈ STRING,
exist ∈ BOOLEAN.

Postconditions : exist est vrai ssi il existe entree ∈ STRING tel que comparer(entree,string) = vrai
si exist est vrai, sortie est la valeur de la sortie correspondant à l'entrée trouvée,
sinon, elle est indéterminée.

c) Module analyse.

Primitive1 : aadresse.

But : vérifier qu'une adresse est structurée de la façon :
local-part "@" domain
c.-à-d. qu'elle vérifie les conditions :
- un seul caractère "@" n'est pas situé entre des guillemets
- le caractère "@" ne peut se trouver en première position
- le caractère "@" ne peut se trouver en dernière position

Arguments : adresse ∈ STRING,
long ∈ ENTIER.

Préconditions : long = longueur(adresse) > 0.

Résultats :
correct,erreur1,erreur2,erreur3,erreur4,erreur5 ∈ BOOLEAN.

Postconditions :
soient les conditions :
C1 : adresse contient au moins un caractère "@",
C2 : un caractère "@" est en première place,
C3 : un caractère "@" est en dernière place,
C4 : le nombre de caractères "@" après le dernier caractère <"> est exactement 1,
C5 : le nombre de caractères "@" après le dernier caractère <"> est > 1 strictement,
C6 : le nombre de caractères "@" après le dernier caractère <"> est égal à 0,
auxquelles correspondent les événements :
erreur1 : pas de caractère "@" dans adresse,
erreur2 : un caractère "@" est en première position,
erreur3 : un caractère "@" est en dernière position,
erreur4 : tout caractère "@" est suivi d'un caractère <">,
erreur5 : plusieurs caractères "@" après le dernier caractère <">.

Les Postconditions sont déterminées par la table :

C1	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
C2	-	Y	N	N	Y	Y	N	N	Y	Y	N
C3	-	N	Y	N	Y	N	Y	N	Y	N	N
C4	-	Y	Y	Y	Y	N	N	N	N	N	N
C5	-	N	N	N	N	Y	Y	Y	Y	N	N
C6	-	N	N	N	N	N	N	N	N	Y	Y
erreur1 = vrai	X										
erreur1 = faux		X	X	X	X	X	X	X	X	X	X
erreur2 = vrai		X			X	X			X	X	
erreur2 = faux	X		X	X			X				X
erreur3 = vrai			X		X		X		X		
erreur3 = faux	X	X		X		X				X	X
erreur4 = vrai										X	X
erreur4 = faux	X	X	X	X	X	X	X		X		
erreur5 = vrai						X	X	X	X		
erreur5 = faux	X	X	X	X	X					X	X
correct = vrai				X							
correct = faux	X	X	X		X	X	X	X	X	X	X

Primitive2 : adomain.

But : vérifier qu'un domaine est structuré de la façon :
domain = domain-syntax *("." domain-syntax)
domain-syntax = alpha [*alphanumhyphen alphanum]
alphanumhyphen = < alpha ou digit ou hyphen >
alpha = < caractère de code décimal ASCII de 65 à 90 et de 97 à 122 >
digit = < caractère de code décimal ASCII de 48 à 57 >
hyphen = < caractère de code décimal ASCII 45 >
c.-à-d. qu'un domaine vérifie les conditions :

- le domaine est formé uniquement de lettres, de chiffres, de traits d'unions et de points,
- le premier caractère du domaine doit être une lettre,
- le dernier caractère ne peut être un point, ni un trait d'union,
- tout caractère suivant un point doit être une lettre,
- tout caractère précédant un point doit être une lettre ou un chiffre,
- deux points ne peuvent se suivre.

Arguments : domaine ϵ STRING, long ϵ ENTIER.

Préconditions : long = longueur(domaine) > 0.

Résultats : correct,erreur1,erreur2,erreur3,erreur4 ϵ BOOLEAN.

Postconditions : soient les conditions :
C1 : premier caractère est une lettre,
C2 : dernier caractère est une lettre ou un chiffre,
C3 : le caractère suivant un point est une lettre,
C4 : le caractère précédant un point est une lettre ou un caractère,
C5 : chaque caractère du domaine est soit une lettre, un chiffre, un point ou un trait d'union,
C6 : deux points se suivent,
auxquelles correspondent les événements :

erreur1 : caractère illégal dans un domaine,
erreur2 : le premier caractère d'un domaine est différent d'une lettre,
erreur3 : le dernier caractère d'un domaine est différent d'une lettre ou d'un chiffre,
erreur4 : un domaine est vide.

Les Postconditions sont déterminées par la table :

C1 et C3	Y	N	Y	Y	Y	N	Y	N	Y	Y	N	Y	N	N	N	N
C2 et C4	Y	Y	N	Y	Y	N	Y	Y	N	N	Y	N	Y	N	N	N
C5	Y	Y	Y	N	Y	Y	N	N	Y	N	Y	N	N	Y	N	N
C6	Y	Y	Y	Y	N	Y	N	Y	N	Y	N	N	N	N	Y	N
erreur1 = V				X			X	X		X		X	X		X	X
erreur1 = F	X	X	X		X	X			X		X			X		
erreur2 = V		X				X		X			X		X	X	X	X
erreur2 = F	X		X	X	X		X		X	X		X				
erreur3 = V			X			X			X	X		X		X	X	X
erreur3 = F	X	X		X	X		X	X			X		X			
erreur4 = V	X	X	X	X		X		X		X					X	
erreur4 = F					X		X		X		X	X	X	X		X
correct = V					X											
correct = F	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X

Primitive3 : aencodedpn.

But : vérifier qu'une local-part est structurée de la façon :
local-part = [given "."] *(initial ".") surname
given = 2*<ps-char not including ".">
initial = alpha
surname = Printablestring
c.-à-d. qu'une local-part vérifie les conditions :
- surname contient un caractère au moins et n'a pas de points dans ces deux premiers caractères,
- le nombre de caractères précédant le premier point est
  - 1 et le caractère est une lettre,
  - > 1 et le caractère "(" ne s'y trouve pas,
- le nombre de caractères entre deux points est
  - 1 et le caractère est
    - une lettre si les deux points n'appartiennent pas à surname,
    - quelconque sinon,
  - > 1 et les caractères sont quelconques,
  - 0 et les deux points appartiennent à surname.

Arguments : local-part ∈ STRING,
long ∈ ENTIER.

Préconditions : long = longueur(encodedpn) > 0,
local-part est en printablestring.

Résultats : correct,erreur1,erreur2,erreur3,erreur4 ∈ BOOLEAN.

Postconditions : soient les conditions :
C1 : un point est présent,
C2 : premier point,
C3 : deuxième caractère de local-part,
C4 : situé après le deuxième caractère de local-part,
C5 : dernier caractère de local-part,
C6 : pas de caractère entre deux points,
C7 : présence d'un caractère "(" avant le point,
C8 : caractère différent d'une lettre,
C9 : le point n'appartient pas à surname,

auxquelles correspondent les événements :

erreur1 : initial incorrect,

erreur2 : given incorrect,

erreur3 : surname incorrect,

erreur4 : pas de surname.

Les Postconditions sont déterminées par la table :

C1 et C2 et C4 et C5	Y	Y	Y	N	N	N	N	Y	Y	Y	N	N	N
C1 et C2 et C3 et C8	N	N	N	Y	Y	Y	N	N	N	N	Y	Y	N
C1 et non C2 et C4 et C9	N	N	N	N	N	-	N	Y	Y	Y	Y	Y	N
C1 et C2 et non C3 et non C4	N	N	N	N	N	N	Y	N	N	N	N	N	N
C1 et C8 et C9	N	N	Y	N	N	Y	N	N	N	Y	N	N	N
C1 et non C2 et C5 et C9	N	Y	N	N	Y	N	N	N	Y	N	N	Y	N
erreur1 = V				X	X	X		X	X	X	X	X	
erreur1 = F	X	X	X				X						X
erreur2 = V	X	X	X					X	X	X			
erreur2 = F				X	X	X	X				X	X	X
erreur3 = V			X			X	X			X			
erreur3 = F	X	X		X	X			X	X		X	X	X
erreur4 = V		X			X				X			X	
erreur4 = F	X		X	X		X	X	X		X	X		X
correct = V													X
correct = F	X	X	X	X	X	X	X	X	X	X	X	X	

Primitive4 : alocalatom.

But : vérifier que local-part est structuré de la façon :
local-part = atom *("." atom)
atom = 1*(CHAR sauf "(", ")", "<", ">", "@", ";", "\",
":", <">, ".", "[", "]" et SPACE et CTLs)
c.-à-d. qu'un local-part vérifie les conditions :
- une local-part ne contient pas de caractères illégaux,
- une local-part ne commence pas par un point,
- une local-part ne se termine pas par un point,
- deux points ne se suivent pas.

Arguments : localpart ∈ STRING,
long ∈ ENTIER.

Préconditions : long = longueur(localpart) > 0.

Résultats :
correct,erreur1,erreur2,erreur3,erreur4 ∈ BOOLEAN.

Postconditions :
soient les conditions :
C1 : localpart contient un caractère illégal,
C2 : le dernier caractère de localpart est un point,
C3 : le premier caractère de localpart est un point,
C4 : deux points se suivent,
auxquelles correspondent les événements :
erreur1 : présence d'un caractère illicite,
erreur2 : local-part se termine par un point,
erreur3 : local-part commence par un point,
erreur4 : absence d'un atom.
On a :
erreur1 est vrai ssi C1 est vrai,
erreur2 est vrai ssi C2 est vrai,
erreur3 est vrai ssi C3 est vrai,
erreur4 est vrai ssi C4 est vrai,
correct est vrai ssi erreur1 est faux et erreur2 est faux
et erreur3 est faux et erreur4 est faux.

Primitive5 : astdorname.

But : vérifier que local-part est structuré de la façon :
local-part = 1*("/" attribute "=" value) "/"
attribute = "C" / "A" / "P" / "X121" / "T-ID" / "O"
/ "OU" / "UA-ID" / "S" / "G" / "I" / "GQ"
/ "PN" / "RFC-822" / "JNT-MAIL" / "UUCP"
value = std-printablestring
std-printablestring = *(std-char / std-pair)
std-char = <ps-delim et ps-char sauf "/" et "=">
std-pair = "(036)" (ps-delim / ps-char)
c.-à-d. que local-part vérifie les conditions :

- elle commence par un caractère "/",
- elle se termine par un caractère "/",
- il existe un seul caractère "=" entre deux caractères "/",
- elle est composée d'attributs existants,
- la valeur associée à l'attribut "PN" est sous forme encoded-pn correcte.

Arguments : local-part ϵ STRING,
long ϵ ENTIER.

Préconditions : long = longueur(std-orname) > 0,
local-part est en printablestring.

Résultats :
correct,erreur1,erreur2,erreur3,erreur4,erreur5 ϵ BOOLEAN.

Postconditions :
soient les conditions :
C1 : le premier caractère de local-part est "/",
C2 : le dernier caractère de local-part est "/",
C3 : un caractère "=" entre deux caractères "/",
C4 : un attribut n'est pas défini dans la syntaxe,
C5 : la valeur de l'attribut "PN" est de syntaxe incorrecte,
auxquelles correspondent les événements :
erreur1 : local-part ne commence pas par "/",
erreur2 : local-part ne se termine pas par "/",
erreur3 : pas ou trop de caractères "=" entre deux caractères "/",
erreur4 : attribut inexistant,
erreur5 : valeur de "PN" de syntaxe incorrecte.

On a :

erreur1 est vrai ssi C1 est faux,
erreur2 est vrai ssi C2 est faux,
erreur3 est vrai ssi C3 est faux,
erreur4 est vrai ssi C4 est vrai,
erreur5 est vrai ssi C5 est vrai,
correct est vrai ssi erreur1 est faux et erreur2 est faux
et erreur3 est faux et erreur4 est faux
et erreur5 est faux.

Primitive6 : alocalword.

But : vérifier qu'une local-part est structurée de la façon :
local-part = word *(".") word
word = atom / quoted-string
quoted-string = <"> * (qtext / quoted-pair) <">
qtext = < CHAR excepté <">, "\", CR >
quoted-pair = "\"
c.-à-d. qu'une local-part vérifie les conditions :

- elle ne commence pas par un caractère ".",
- elle ne se termine pas par un caractère ".",
- il y a un nombre pair d'occurrences du symbole <">
non précédé du symbole "\",
- toute occurrence impaire du symbole <"> est précédée
du symbole "." sauf si elle est en première position.
Seuls sont comptabilisés les caractères <"> non précédés
du caractère "\",
- toute occurrence paire du symbole <"> est suivie du
symbole "." sauf s'il est en dernière position. Seuls
sont comptabilisés les caractères <"> non précédés
du caractère "\",
- chaque atom ne contient pas de caractères illégaux,
- deux points se suivent seulement dans un quoted-string.

```
Arguments : localpart ∈ STRING,
            long ∈ ENTIER.

Préconditions : long = longueur(localpart) > 0.

Résultats :
  correct,erreur1,erreur2,erreur3 ∈ BOOLEAN.

Postconditions :
  soient les conditions :
C1 : un point est en première position,
C2 : un point est en dernière position,
C3 : un caractère illégal dans un atom,
C4 : deux points se suivent hors d'un quoted-string,
C5 : il existe une occurrence impaire d'un symbole <">
     non située en première position qui n'est pas
     précédée d'un point,
C6 : il existe une occurrence paire d'un symbole <"> non
     située en dernière position qui n'est pas suivie
     d'un point,
C7 : un nombre impair de symboles <"> non précédés
     d'un caractère "\",
  auxquelles correspondent les événements :
erreur1 : word non présent,
erreur2 : syntaxe d'atom incorrecte,
erreur3 : quoted-string non borné.
```

Les Postconditions sont déterminées par la table :

C1 ou C2 ou C4	N	Y	N	N	Y	N	Y	Y
C3 ou C5 ou C6	N	N	Y	N	N	Y	Y	Y
C7	N	N	N	Y	Y	Y	N	Y
erreur1 = Vrai		X			X		X	X
erreur1 = Faux	X		X	X		X		
erreur2 = Vrai			X			X	X	X
erreur2 = Faux	X	X		X	X			
erreur3 = Vrai				X	X	X		X
erreur3 = Faux	X	X	X				X	
correct = Vrai	X							
correct = Faux		X	X	X	X	X	X	X

Primitive7 : aroute.

But : vérifier qu'une adresse est structurée de la façon :
adresse = route ":" local-part "@" domain
route = 1#("@" domain)
c.-à-d. que adresse vérifie les conditions :

- domain est de syntaxe correcte,
- local-part "@" domain est de syntaxe correcte,
- tout domaine de route commence par le caractère "@" et se termine par le caractère "," ou plusieurs caractères ",' excepté pour le dernier,
- le premier caractère de local-part "@" domain est le premier caractère suivant le caractère ":".

```
Arguments : adresse ∈ STRING,
            long ∈ ENTIER.

Préconditions : long = longueur(adresse) > 0,
                caractere(adresse,1) = "@".

Résultats :
  correct,erreur1,erreur2,erreur3,erreur4,erreur5,erreur6,
  erreur7,erreur8 ∈ BOOLEAN.

Postconditions :
  soient les conditions :
C1 : adresse contient le caractère ":",
C2 : le caractère ":" est en dernière position,
C3 : la partie local-part"@"domain est de syntaxe correcte,
C4 : la partie domain de l'adresse spécifique est de syntaxe
     correcte,
C5 : la partie local-part de l'adresse spécifique est de
     syntaxe correcte,
C6 : pas de caractères entre le caractère "@" et le caractère
     ":",
C7 : le caractère ":" est en deuxième position dans adresse,
C8 : un domaine de route est de syntaxe correcte,
C9 : pas de caractère "," à la fin d'un domaine de route qui
     n'est pas le dernier,
  auxquelles correspondent les événements :
erreur1 : pas de caractère ":",
erreur2 : absence de la partie local-part "@" domain,
erreur3 : la syntaxe local-part "@" domain est erronée,
erreur4 : la syntaxe du domaine de l'adresse spécifique
          est erronée,
erreur5 : la syntaxe de local-part est erronée,
erreur6 : absence de la partie domain dans route,
erreur7 : la syntaxe d'un domaine dans route est erronée,
erreur8 : absence du caractère ",".
```

Les Postconditions sont déterminées par les tables :

C1	N	Y	Y	Y	Y	Y	Y
C2	-	Y	N	N	N	N	N
C3	-	-	N	Y	Y	Y	Y
C4	-	-	-	N	N	Y	Y
C3	-	-	-	N	Y	N	Y
erreur1 = Vrai	X						
erreur1 = Faux		X	X	X	X	X	X
erreur2 = Vrai		X					
erreur2 = Faux	X		X	X	X	X	X
erreur3 = Vrai			X				
erreur3 = Faux	X	X		X	X	X	X
erreur4 = Vrai				X	X		
erreur4 = Faux	X	X	X			X	X
erreur5 = Vrai				X		X	
erreur5 = Faux	X	X	X		X		X
correct1 = Vrai							X
correct1 = Faux	X	X	X	X	X	X	

C1	N	Y	Y	Y	Y	Y	Y	Y	Y
C6	-	-	Y	N	Y	N	N	Y	N	Y
C7	-	Y	N	N	N	N	N	N	N	N
C8	-	-	Y	N	N	Y	N	Y	Y	N
C9	-	-	Y	N	N	N	Y	N	Y	Y
erreur1 = vrai	X									
erreur1 = faux		X	X	X	X	X	X	X	X	X
erreur6 = vrai		X	X		X			X		X
erreur6 = faux	X			X		X	X		X	
erreur7 = vrai				X	X		X			X
erreur8 = faux	X	X	X			X		X	X	
correct2 = vrai						X				
correct2 = faux	X	X	X	X	X		X	X	X	X

correct est vrai ssi correct1 est vrai et correct2 est vrai.

Primitive8 : aattribut.

But : déterminer dans les adresses cibles d'une table de conversion la présence des attributs standards de X400 : CountryName, AdministrationDomainName, privateDomainName, OrganizationName.

Arguments : sortie ∈ STRING,
long ∈ ENTIER.

Préconditions : long = longueur(sortie) > 0,
sortie est sous la forme :
attribute "=" value *(";" *(SPACE) attribute "=" value).

Résultats : C, A, P, O ∈ BOOLEAN.

Postconditions : C est vrai ssi "C=" est dans sortie,
A est vrai ssi "A=" est dans sortie,
P est vrai ssi "P=" est dans sortie,
O est vrai ssi "O=" est dans sortie.

Primitive9 : aformat.

But : vérifier que les attributs constituant un élément de type STRING représente un ORName correct. Dans le cas où les attributs "C" et "A" sont absents et l'adresse RFC-822 d'origine spécifie la passerelle, l'ORName est complété avec les attributs de la passerelle.

Arguments : string1 ∈ STRING,
table ∈ STRING,
passerelle ∈ BOOLEAN,
long ∈ ENTIER.

Preconditions : string1 est sous la forme :
attribute "=" value *(";" *(SPACE) attribute "=" value)
attribute existe,
table existe ssi passerelle est vrai.

Résultats : string2 ∈ STRING,
correct,erreur1,erreur2,erreur3 ∈ BOOLEAN.

Postconditions :
soient les conditions :
C1 : string1 contient deux attributs semblables,
C2 : passerelle est vrai et attribut "C" et "A" absents dans string1,
C3 : les attributs ne constituent pas une forme de ORName correcte,
C4 : la valeur d'un attribut est erronée,
auxquelles correspondent les événements :
erreur1 : deux attributs identiques,
erreur2 : forme de ORName erronée,
erreur3 : valeur d'un attribut erronée,
result1 : string2 = string1,
result2 : string2 = string1 complété des attributs "C" et "A" de la passerelle.

Les Postconditions sont déterminées par la table ;

C1	Y	N	Y	Y	N	N	N	Y	N
C2	-	Y	-	-	-	-	-	-	N
C3	Y	N	Y	N	Y	N	Y	N	N
C4	Y	N	N	Y	Y	Y	N	N	N
erreur1 = vrai	X		X	X				X	
erreur1 = faux		X			X	X	X		X
erreur2 = vrai	X		X		X		X		
erreur2 = faux		X		X		X		X	X
erreur3 = vrai	X			X	X	X			
erreur3 = faux		X	X				X	X	X
result1	X		X	X	X	X	X	X	X
result2		X							
correct = vrai		X							X
correct = faux	X		X	X	X	X	X	X	

d) Module MAPPING.

Primitive1 : mapasciipr.

But : transformer une local-part formée de caractères ASCII dans le sous-ensemble Printablestring suivant la correspondance de ASCII vers Printablestring définie au chapitre IV.

Arguments : local-part ϵ STRING.

Préconditions : longueur(local-part) > 0.

Résultats : printable ϵ STRING,
erreur ϵ BOOLEAN.

Postconditions :
printable = local-part avec les transformations :
tout caractère "@" se transforme en "(a)",
tout caractère "%" se transforme en "(p)",
tout caractère "!" se transforme en "(b)",
tout caractère <"> se transforme en "(q)",
tout caractère "_" se transforme en "(u)",
tout autre caractère différent de ps-char et de ps-delim se transforme en "("3DIGIT")" qui représente le code décimal ASCII du caractère
et
erreur = faux
ou
printable = local-part
et
erreur = vrai si la transformation d'un caractère n'a pu se faire.

Primitive2 : maplocascii.

But : transformer une local-part selon la correspondance
local-part vers ASCII définie au chapitre IV.

Arguments : local-part ϵ STRING.

Préconditions : longueur(local-part) > 0.

Résultat : ascii ϵ STRING,
erreur ϵ BOOLEAN.

Postconditions :
si local-part a la structure :
local-part = atom-encoded *("." atom-encoded)
atom-encoded = * (a-char / a-encoded-char)
a-char = < CHAR sauf "(", ")", "<", ">", "@", ",", ";",
":", "\", <">, ".", "[", "]", SPACE, CTL, "-", "#" >
a-encoded-char = "_" / "#u#" / "#l#" / "#r#" / "#m#" /
"#c#" / "#b#" / "#h#" / "#e#" / "#s#" / "#"3DIGIT"#",
ascii = local-part avec les transformations :
"." se transforme en ".",
"_" se transforme en space,
"#u#" se transforme en "_",
"#l#" se transforme en "(",
"#r#" se transforme en ")",
"#m#" se transforme en ",",
"#c#" se transforme en ":",
"#b#" se transforme en "\",
"#h#" se transforme en "#",
"#e#" se transforme en "$=",
"#s#" se transforme en "$/",
"#"3DIGIT"#" se transforme en caractère de code décimal
ASCII déterminé par les 3 chiffres.
Sinon, ascii = local-part.
ereur = vrai ssi la transformation d'un caractère n'a pu se faire.

Primitive3 : mapdomX400.

But : transformer un domaine RFC-822 selon la correspondance de base contrôlée par les exceptions, définie au chapitre IV.

Arguments : domain ϵ STRING.

Préconditions : domain est syntaxiquement correct,
longueur(domain) > 0.

Résultats : listeattributs ϵ STRING.

Postconditions :
listeattributs est la transformation de domain selon la correspondance de base contrôlée par les exceptions.
listeattributs est structuré de la façon :
listeattributs =
domain-syntax *(";" *(SPACE) attribut "=" domain-syntax)
attribut = "C" / "A" / "P" / "O" / "OU".

Primitive4 : mapencoded.

But : transformer une local-part sous forme encoded-pn selon la correspondance local-part vers PersonalName définie au chapitre IV.

Arguments : local-part ϵ STRING.

Préconditions : longueur(local-part) > 0,
local-part est en syntaxe encoded-pn correcte.

Résultat : x400 ϵ STRING.

Postconditions :
x400 = 1 * (attribut = value ";")
attribut = "S" / "GI" / "I".

Primitive5 : maporname.

But : transformer une local-part en stdorname sous la forme :
1*(attribute "=" value ";") avec l'attribut "PN" encodé
selon la transformation mapencoded.

Argument : local-part ϵ STRING.

Préconditions : longueur(local-part) > 0,
local-part est en syntaxe std-orname correcte.

Résultats : x400 ϵ STRING.

Postconditions :
local-part est sous la forme :
local-part = 1 * (attribute = value ";")
et si attribute = "PN", value est encodé selon la
transformation mapencoded.

e) Module ERREUR.

Primitive1 : error.

But : Donner l'erreur correspondant à un numéro.

Arguments : numero ∈ ENTIER.

Préconditions : numero existe.

Résultats : erreur ∈ STRING.

Postconditions :
numero = 1 : absence d'un caractère "@" dans l'adresse,
numero = 2 : un caractère "@" est en dernière position
dans l'adresse,
numero = 3 : pas de caractère "@" pour déterminer le domaine,
numero = 4 : plusieurs caractères "@" déterminent une route
mais aucun est en première position,
numero = 5 : existence d'un caractère différent d'une lettre,
d'un chiffre, d'un trait d'union ou d'un point
dans la partie domaine de l'adresse,
numero = 6 : le premier caractère d'un sous-domaine est
différent d'une lettre,
numero = 7 : le dernier caractère d'un sous-domaine est
différent d'une lettre ou d'un chiffre,
numero = 8 : absence d'un sous-domaine,
numero = 9 : pas de correspondance d'un caractère ASCII en
printablestring,
numero = 10: si la partie locale est en syntaxe std-orname,
le premier caractère doit être "/",
numero = 11: si la partie locale est en syntaxe std-orname,
le dernier caractère doit être "/",
numero = 12: si la partie locale est en syntaxe std-orname,
un seul caractère "=" doit se trouver entre deux
"/",
numero = 13: si la partie locale est en syntaxe std-orname,
un attribut n'est pas défini,

numero = 14: si la partie locale est en syntaxe std-orname, la valeur de l'attribut "PN" est incorrecte,

numero = 15: si la partie locale est en syntaxe encoded-pn, le terme initial est incorrect,

numero = 16: si la partie locale est en syntaxe encoded-pn, le terme givenname est incorrect,

numero = 17: si la partie locale est en syntaxe encoded-pn, le terme surname est incorrect,

numero = 18: si la partie locale est en syntaxe encoded-pn, absence d'un surname,

numero = 19: ORName avec deux attributs identiques,

numero = 20: forme incorrecte d'ORName,

numero = 21: valeur incorrecte d'attribut dans l'ORName,

numero = 22: pas d'attributs spécifiant la passerelle,

numero = 23: absence du caractère ":" pour délimiter la route,

numero = 24: absence de la partie adresse spécifique dans l'adresse,

numero = 25: la syntaxe de l'adresse spécifique est erronée,

numero = 26: la syntaxe du domaine de l'adresse spécifique est erronée,

numero = 27: la syntaxe de la partie locale de l'adresse spécifique est erronée,

numero = 28: absence d'un domaine dans la partie route,

numero = 29: la syntaxe d'un domaine dans la partie route est erronée,

numero = 30: absence d'un caractère "," à la fin d'un domaine de la partie route,

numero = 31: un 822-P1 Recipient sous forme route-address est rejeté.

IV) Intégration de la passerelle.

Notre travail d'implémentation a donc été de programmer une passerelle limitée au problème des adresses dans le sens RFC-822 vers X400, sous le système d'exploitation UNIX. Ce programme doit être inséré dans un site gérant un MTAE et un noeud du monde RFC-822. Le système de courrier électronique des Facultés Notre-Dame de la Paix à Namur dispose de ces deux éléments et d'un gestionnaire de messages entre plusieurs courriers électroniques, de syntaxe différente. Ce programme appelé SENDMAIL dirige un message vers le type de courrier souhaité [Allman E., Sendmail]. Son but est d'effectuer une transition entre des systèmes de transfert de messages ayant un mode d'adressage différent; il n'a aucun interface avec les utilisateurs et ne livre pas les messages. La configuration d'une application utilisant SENDMAIL est la suivante (cfr figure VI.2) :

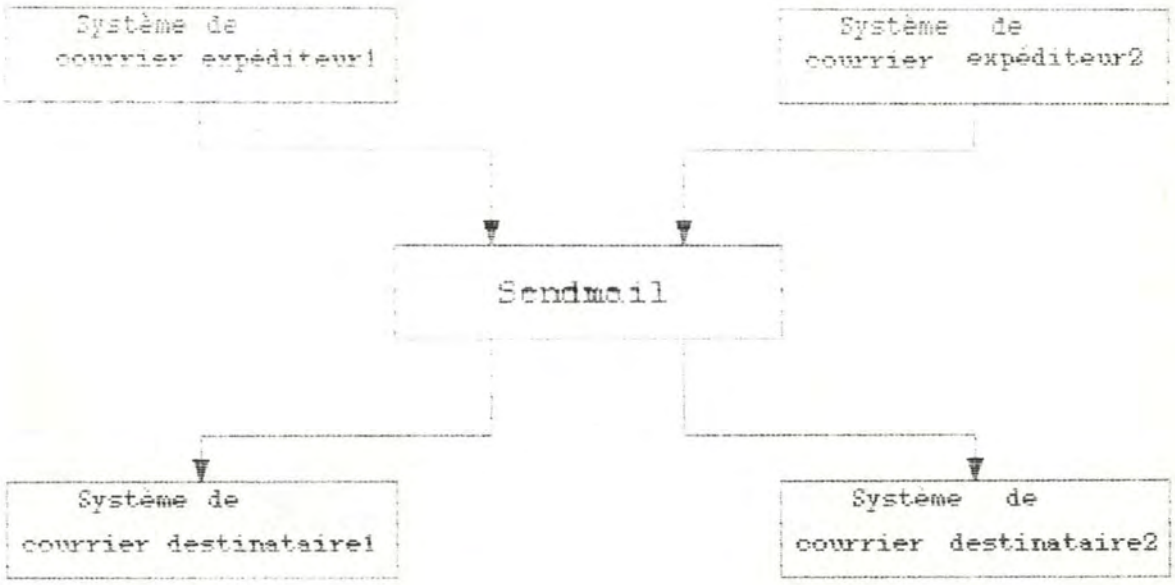


FIGURE VI.2. : CONFIGURATION D'UTILISATION DE SENDMAIL

SENDMAIL est le lien entre un système de transfert qui reçoit un message et un autre système qui le livrera aux utilisateurs souhaités. Eric Allman a conçu ce programme dans l'optique d'interconnecter les courriers électroniques qui respectent des variantes de la syntaxe RFC-822. L'interprétation des adresses est contrôlée par un ensemble de règles de réecriture. Ces règles analysent les adresses du service 822-P1 et de l'en-tête du message, les transforment dans le format exigé par le courrier électronique destinataire.

Cinq ensembles de règles de réécriture existent, ayant une sémantique spécifique (cfr figure VI.3.) :

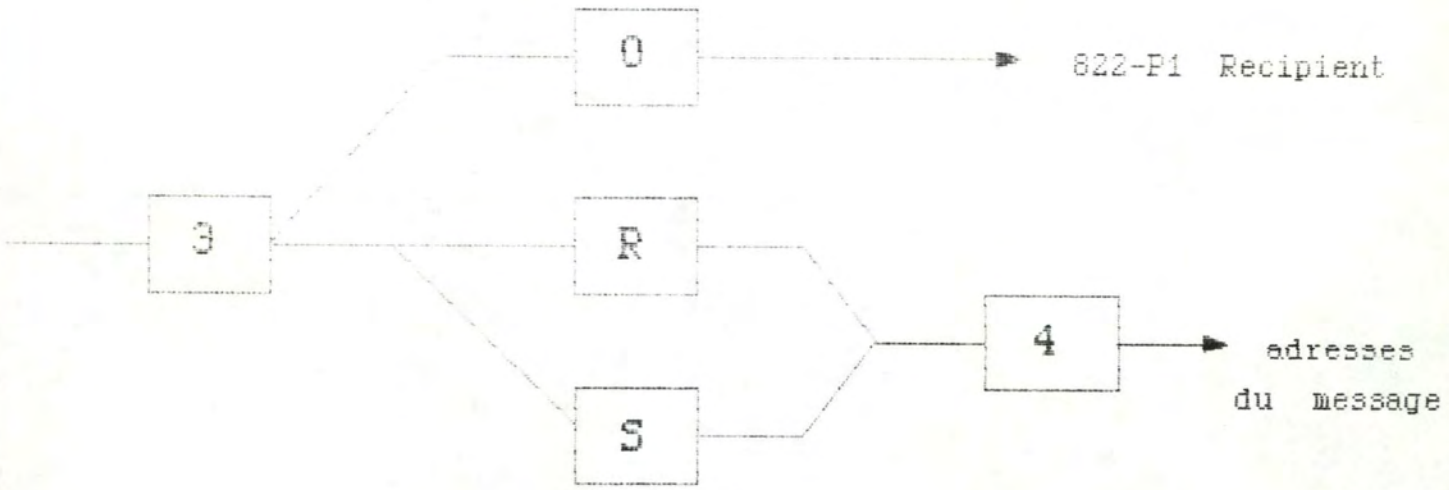


FIGURE VI.3. : SEMANTIQUE DES ENSEMBLES DE REGLES DE REECRITURE

- l'ensemble des règles 3 transforme toute adresse en format RFC-822 pur :

local-part @ domain.

- l'ensemble des règles 0 est appliqué après les règles 3 sur l'adresse 822-P1 Recipient et interpréte le nom du courrier électronique destinataire.

- les ensembles de règles R, S et 4 sont appliqués sur les adresses des destinataires (R) et de l'expéditeur (S) pour être transformées dans le format d'adresse spécifié par le courrier électronique destinataire.

SENDMAIL est paramétré par un fichier de configuration qui définit, en outre, les systèmes de courrier électronique destinataires et les ensembles de règles de réécriture d'adresses.

Nous avons modifié ce fichier de configuration pour que SENDMAIL transmette à notre passerelle les messages venant du monde RFC-822 et dont la destinatation est un site X400 :

nous avons ajouté la définition d'un nouveau système de courrier électronique qui spécifie le chemin et le nom de notre passerelle et nous avons modifié l'ensemble des règles 0 pour qu'il sélectionne les messages à destination X400.

V. Tests et résultats.

La résolution des exemples présentés ci-dessous est basée sur la table de correspondance :

```
rtt.be          #A=rtt; C=be
info.rtt.be     #D=info; P=fndp; A=rtt; C=be
dbp.de          #D=; A=dbp; C=de
mpg.dbp.de      #P=mpg; A=dbp; C=de
fr              #P=fnet; A=PTT; C=FR
cea.ptt.fr      #D=cea; P=aristote; A=PTT; C=FR
```

Le programme est testé d'une part avec Sendmail et d'autre part de façon isolée. Ces tests ne sont pas exhaustifs mais ils donnent un aperçu général de la passerelle.

a) Par Sendmail.

Sendmail est configuré comme suit :

sont dirigés vers notre passerelle tous les messages dont l'adresse destinataire est de la forme :

```
local-part@domain1.domain2.rtt.be
local-part@domain1.domain2.dbp.de
local-part@domain1.domain2.ptt.fr
local-part@domain1.domain2.rtt.be
local-part@domain1.domain2.dbp.de
local-part@domain1.domain2.ptt.fr
```

```
Date: Fri, 26 Aug 88 16:04:31 GMT
From: student <vfa@alma.UUCP>
Message-Id: <8808261604.AA01311@alma.UUCP>
Received: by alma.UUCP id AA01311; Fri, 26 Aug 88 16:04:31 GMT
To: M.Rose@info.rtt.be, Rose@IIHE.rtt.be
Subject: test passerelle
Cc: Marshall.Rose@bio.fhg.dbp.de

TEST 1


 CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
 --------------------------------------------

      adresse en format rfc-822 : M.Rose@info.rtt.be

      adresse en format X400 : I=M;
                               S=Rose;
                               O=info;
                               P=fndp;
                               A=rtt;
                               C=be


 CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
 --------------------------------------------

      adresse en format rfc-822 : Marshall.Rose@bio.fhg.dbp.de

      adresse en format X400 : GI=Marshall;
                               S=Rose;
                               OU=bio;
                               O=;
                               P=fhg;
                               A=dbp;
                               C=de


 CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
 --------------------------------------------

      adresse en format rfc-822 : Rose@IIHE.rtt.be

      adresse en format X400 : S=Rose;
                               P=IIHE;
                               A=rtt;
                               C=be
```

```
Date: Fri, 26 Aug 88 16:10:18 GMT
From: student <vfa@alma.UUCP>
Message-Id: <8808261610.AA01335@alma.UUCP>
Received: by alma.UUCP id AA01335; Fri, 26 Aug 88 16:10:18 GMT
To: Marshall.MT@passerelle.rtt.be
Subject: test passerelle
Cc: /PN=Marshall.M.T.Rose/GQ=5/@cea.ptt.fr

TEST2
```

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400

adresse en format rfc-822 : /PN=Marshall.M.T.Rose/GQ=5/@cea.ptt.fr

adresse en format X400 : GI=Marshall;
I=MT;
S=Rose;
GQ=5;
O=cea;
P=aristote;
A=PTT;
C=FR

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400

adresse en format rfc-822 : Marshall.MT@passerelle.rtt.be

adresse en format X400 : GI=Marshall;
S=MT;
A=rtt;
C=be;

```
Date: Fri, 26 Aug 88 16:15:39 GMT
From: student <vfa@alma.UUCP>
Message-Id: <8808261615.AA01358@alma.UUCP>
Received: by alma.UUCP id AA01358; Fri, 26 Aug 88 16:15:39 GMT
To: Marshall.MT.Rose@passerelle.rtt.be
Subject: test passerelle
Cc: /C=be/A=rtt/P=kul-cs/@passerelle.rtt.be

TEST 3
```

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400

adresse en format rfc-822 : /C=be/A=rtt/P=kul-cs/@passerelle.rtt.be

adresse en format X400 : P=kul-cs;
A=rtt;
C=be;

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400

adresse en format rfc-822 : Marshall.MT.Rose@passerelle.rtt.be

adresse en format X400 : GI=Marshall;
S=MT.Rose;
A=rtt;
C=be;

```
Date: Fri, 26 Aug 88 16:19:38 GMT
From: student <vfa@alma.UUCP>
Message-Id: <8808261619.AA01376@alma.UUCP>
Received: by alma.UUCP id AA01376; Fri, 26 Aug 88 16:19:38 GMT
To: "/C=de/A=dbp/RFC-822=Postel@usc-isib.arpa/"@passerelle.rtt.be

TEST 4
```

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400

adresse en format rfc-822 :

"/C=de/A=dbp/RFC-822=Postel@usc-isib.arpa/"@passerelle.rtt.be

adresse en format X400 : RFC-822=Postel(a)usc-isib.arpa;
A=dbp;
C=de;

```
Date: Sun, 28 Aug 88 23:45:53 GMT
From: student <vfa@alma.UUCP>
Message-Id: <8808282345.AA01759@alma.UUCP>
Received: by alma.UUCP id AA01759; Sun, 28 Aug 88 23:45:53 GMT
To: Marshall..M.Rose@info..rtt.be
Subject: test erreur
Cc: /Cde/A=dbp/@passerell-.rtt.be

TEST 5
```

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
---

adresse en format rfc-822 : /Cde/A=dbp/@passerell-.rtt.be

ERREURS :

le dernier caractere d'un sous-domaine est different
d'une lettre ou d'un chiffre
si la partie locale est en syntaxe std-orname,
un seul caractere = doit se trouver entre deux /

adresse en format X400 :

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
---

adresse en format rfc-822 : Marshall..M.Rose@info..rtt.be

ERREURS :

absence d'un sous-domaine
si la partie locale est en syntaxe std-orname,
le premier caractere doit etre /
si la partie locale est en syntaxe std-orname,
le dernier caractere doit etre /
si la partie locale est en syntaxe encoded-pn,
le terme surname est incorrect

adresse en format X400 :

```
Date: Sun, 28 Aug 88 23:54:28 GMT
From: student <vfa@alma.UUCP>
Message-Id: <8808282354.AA01780@alma.UUCP>
Received: by alma.UUCP id AA01780; Sun, 28 Aug 88 23:54:28 GMT
To: /PNN=M.rose/@lea.rtt.be
Subject: test erreur
Cc: /C=be/C=dbp/@passerelle.rtt.be

TEST 6
```

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
---

adresse en format rfc-822 : /C=be/C=dbp/@passerelle.rtt.be

ERREURS :

ORName avec deux attributs identiques
forme incorrecte d'ORName

adresse en format X400 :

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
---

adresse en format rfc-822 : /PNN=M.rose/@lea.rtt.be

ERREURS :

le premier caractere d'un sous-domaine est different d'une lettre
si la partie locale est en syntaxe std-orname,
un attribut n'est pas defini

adresse en format X400 :

b) De façon isolée.

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400

adresse en format rfc-822 : Marshall.M.T.Rose@inria.fr

adresse en format X400 : GI=Marshall;
I=MT;
S=Rose;
O=inria;
P=fnet;
A=PTT;
C=FR

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400

adresse en format rfc-822 : @hostx,@hosty:Smith@rtt.be

adresse en format X400 : RFC-822(a)hostx,(a)hosty:Smith(a)rtt.be
A=rtt;
C=be;

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400

adresse en format rfc-822 : Marshall.MT.Rose@ptt.at

adresse en format X400 : GI=Marshall;
S=MT.Rose;
A=ptt;
C=at;

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
---

adresse en format rfc-822 : @hostx,@hostY:Smith@rtt.be

ERREURS :

un 822-P1 Recipient sous forme route-address est rejete

adresse en format X400 :

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
---

adresse en format rfc-822 : vfa

ERREURS :

absence d'un caractere @ dans l'adresse

adresse en format X400 :

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
---

adresse en format rfc-822 : hostx,@nosty:smith@rtt.be

ERREURS :

plusieurs caracteres @ determinent une route
mais aucun n'est en premiere position

adresse en format X400 :

CORRESPONDANCE D'UNE ADRESSE RFC-822 EN X400
---

adresse en format rfc-822 : "postel@usc.arpa"rtt.be

ERREURS :

pas de caractere @ pour determiner le domaine
plusieurs caracteres @ determinent une route
mais aucun n'est en premiere position

adresse en format X400 :

VII.CONCLUSION.

Les systèmes proposant des services de traitement de messages connaissent actuellement un vif succès : la messagerie électronique, nouvel outil de communication alliant les moyens informatiques et les possibilités offertes par les réseaux, permet à ses utilisateurs d'échanger des messages de tout type. Divers comités ont donc développé leurs propres normes, engendrant une incapacité de communiquer entre les différents systèmes de courrier. Dans l'optique d'établir une messagerie électronique mondiale, le CCITT (Comité Consultatif International de Télégraphie et de Téléphonie) a publié en 1984 les Recommandations X400. Une solution à moyen terme pour le problème de la migration vers les systèmes de traitement de messages se conformant aux normes X400 est offerte par l'établissement de passerelles de courrier

Notre travail s'est axé sur l'analyse du document RFC-987 de Steve Kille qui propose une spécification d'interconnexion entre un monde RFC-822 et un monde X400. L'établissement d'une telle passerelle se doit d'être la plus générale possible mais la difficulté de compréhension de ces normes extrêmement techniques nous a conduit à poser des hypothèses sur la correspondance entre les adresses et entre les services. Ces restrictions donnent une application personnelle pratique qui reste malgré tout assez générale. L'implémentation de la conversion d'une adresse de format RFC-822 en ORName X400 nous a permis d'entrevoir la lourde tâche que nécessite la réalisation physique d'une passerelle générale , celle-ci étant laissée à une éventuelle continuation de ce mémoire.

ANNEXE 1 : LE PROTOCOLE P1 ET P2 DES RECOMMANDATIONS X400

```
P1 DEFINITIONS ::=
BEGIN
-- P1 makes use of types defined in the following module
-- T73: Recommendation T.73
MPDU                    ::= CHOICE {[0] IMPLICIT User MPDU, Service MPDU}
ServiceMPDU             ::= CHOICE {[1] IMPLICIT DeliveryReportMPDU,
                                    [2] IMPLICIT ProbeMPDU}
UserMPDU                ::= SEQUENCE {UMPDUEnvelope, UMPDUContent}
UMPDUEnvelope           ::= SET {
                              MPDUIdentifier,
                              originator ORName,
                              original EncodedInformationTypes OPTIONAL,
                              ContentType,
                              UAContentID OPTIONAL,
                              Priority DEFAULT normal,
                              PerMessageFlag DEFAULT {},
                              deferredDelivery [0] IMPLICIT Time OPTIONAL,
                              [1] IMPLICIT SEQUENCE OF PerDomainBilateralInfo OPTIONAL,
                              [2] IMPLICIT SEQUENCE OF RecipientInfo,
                              TraceInformation}
UMPDUContent            ::= OCTET STRING
-- time
Time                    ::= UTCTime
-- various envelope information
MPDUIdentifier          ::= [APPLICATION 4] IMPLICIT SEQUENCE {
                              GlobalDomainIdentifier, IA5String}
ContentType             ::= [APPLICATION 6] IMPLICIT INTEGER {p2(2)}
UAContentID             ::= [APPLICATION 10] IMPLICIT PrintableString
Priority                ::= [APPLICATION 7] IMPLICIT INTEGER {
                              normal(0), nonUrgent(1), urgent(2)}
PerMessageFlag          ::= [APPLICATION 8] IMPLICIT BIT STRING {
                              discloseRecipients(0),
                              conversionProhibited(1),
                              alternateRecipientAllowed(2),
                              contentReturnRequest(3)}
-- per-domain bilateral information
PerDomainBilateralInfo  ::= SEQUENCE {
                              CountryName,
                              AdministrationDomainName,
                              BilateralInfo}
```

FIGURE 16/X.411 (Part 1 of 4)

Formal definition of User MPDU

```
-- P1 DEFINITIONS continued --
BilateralInfo                ::= ANY
-- recipient information
RecipientInfo                ::= SET {
                                     recipient ORName,
                                     [0] IMPLICIT ExtensionIdentifier,
                                     [1] IMPLICIT PerRecipientFlag,
                                     [2] IMPLICIT ExplicitConversion OPTIONAL}
ExtensionIdentifier          ::= INTEGER
PerRecipientFlag             ::= BIT STRING -- see Figure 19/X.411
ExplicitConversion           ::= INTEGER {iA5TextTeletex(0), teletexTelex(1)}
-- trace information
TraceInformation             ::= [APPLICATION 9] IMPLICIT SEQUENCE OF
                                     SEQUENCE {GlobalDomainIdentifier, DomainSuppliedInfo}
DomainSuppliedInfo           ::= SET {
                                     arrival [0] IMPLICIT Time,
                                     deferred [1] IMPLICIT Time OPTIONAL,
                                     action [2] IMPLICIT INTEGER {relayed(0), rerouted(1)},
                                     converted EncodedInformationTypes OPTIONAL,
                                     previous GlobalDomainIdentifier OPTIONAL}
-- global domain identifier
GlobalDomainIdentifier       ::= [APPLICATION 3] IMPLICIT SEQUENCE {
                                     CountryName,
                                     AdministrationDomainName,
                                     PrivateDomainIdentifier OPTIONAL}
CountryName                  ::= [APPLICATION 1] CHOICE {
                                     NumericString,
                                     PrintableString}
AdministrationDomainName     ::= [APPLICATION 2] CHOICE {
                                     NumericString,
                                     PrintableString}
PrivateDomainIdentifier      ::= CHOICE {
                                     NumericString,
                                     PrintableString}
-- O/R name
ORName                       ::= [APPLICATION 0] IMPLICIT SEQUENCE {
                                     StandardAttributeList,
                                     DomainDefinedAttributeList OPTIONAL}
```

FIGURE 16/X.411 (Part 2 of 4)

Formal definition of User MPDU

```
-- P1 DEFINITIONS continued --
StandardAttributeList          ::= SEQUENCE {
                                       CountryName OPTIONAL,
                                       AdministrationDomainName OPTIONAL,
                                       [0] IMPLICIT X121Address OPTIONAL,
                                       [1] IMPLICIT TerminalID OPTIONAL,
                                       [2] PrivateDomainName OPTIONAL,
                                       [3] IMPLICIT OrganizationName OPTIONAL,
                                       [4] IMPLICIT UniqueUAIdentifier OPTIONAL,
                                       [5] IMPLICIT PersonalName OPTIONAL,
                                       [6] IMPLICIT SEQUENCE OF OrganizationalUnit OPTIONAL}
DomainDefinedAttributeList     ::= SEQUENCE OF DomainDefinedAttribute
DomainDefinedAttribute         ::= SEQUENCE {
                                       type PrintableString,
                                       value PrintableString}
X121Address                    ::= NumericString
TerminalID                     ::= PrintableString
OrganizationName               ::= PrintableString
UniqueUAIdentifier             ::= NumericString
PersonalName                   ::= SET {
                                       surName [0] IMPLICIT PrintableString,
                                       givenName [1] IMPLICIT PrintableString OPTIONAL,
                                       initials [2] IMPLICIT PrintableString OPTIONAL,
                                       generationQualifier [3] IMPLICIT PrintableString OPTIONAL}
OrganizationalUnit             ::= PrintableString
PrivateDomainName              ::= CHOICE {
                                       NumericString,
                                       PrintableString}
-- encoded information types
EncodedInformationTypes        ::= [APPLICATION 5] IMPLICIT SET {
                                       [0] IMPLICIT BIT STRING {
                                           undefined(0), tLX(1), iA5Text(2), g3Fax(3),
                                           tIF0(4), tTX(5), videotex(6), voice(7), sFD(8),
                                           tIF1(9)},
                                       [1] IMPLICIT G3NonBasicParams OPTIONAL,
                                       [2] IMPLICIT TeletexNonBasicParams OPTIONAL,
                                       [3] IMPLICIT PresentationCapabilities OPTIONAL
                                       -- other non-basic parameters are for further study --}
```

FIGURE 16/X.411 (Part 3 of 4)

Formal definition of User MPDU

```
-- P1 DEFINITIONS continued --
NonBasicParams          ::= BIT STRING {
                              twoDimensional(8),
                              fineResolution(9),
                              unlimitedLength(20),
                              b4Length(21),
                              a3Width(22),
                              b4Width(23),
                              uncompressed(30)}
letexNonBasicParams     ::= SET {
                              graphicCharacterSets [0] IMPLICIT T61String OPTIONAL,
                              controlCharacterSets [1] IMPLICIT T61String OPTIONAL,
                              pageFormats [2] IMPLICIT OCTET STRING OPTIONAL,
                              miscTerminalCapabilities [3] IMPLICIT T61String OPTIONAL,
                              privateUse [4] IMPLICIT OCTET STRING OPTIONAL}
esentationCapabilities  ::= T73.PresentationCapabilities
```

FIGURE 16/X.411 (Part 4 of 4)

Formal definition of User MPDU

```
-- P1 DEFINITIONS continued --
DeliveryReportMPDU          ::= SEQUENCE {
                                    DeliveryReportEnvelope, DeliveryReportContent}
DeliveryReportEnvelope      ::= SET {
                                    report MPDUIdentifier,
                                    originator ORName,
                                    TraceInformation}
DeliveryReportContent       ::= SET {
                                    original MPDUIdentifier,
                                    intermediate TraceInformation OPTIONAL,
                                    UAContentId OPTIONAL,
                                    [0] IMPLICIT SEQUENCE OF ReportedRecipientInfo,
                                    returned [1] IMPLICIT UMPDUContent OPTIONAL,
                                    billingInformation [2] ANY OPTIONAL}
ReportedRecipientInfo       ::= SET {
                                    recipient [0] IMPLICIT ORName,
                                    [1] IMPLICIT ExtensionIdentifier,
                                    [2] IMPLICIT PerRecipientFlag,
                                    [3] IMPLICIT LastTraceInformation, intendedRecipient
                                    [4] IMPLICIT ORName OPTIONAL,
                                    [5] IMPLICIT SupplementaryInformation OPTIONAL}
-- last trace information
LastTraceInformation        ::= SET {
                                    arrival [0] IMPLICIT Time,
                                    converted EncodedInformationTypes OPTIONAL,
                                    [1] Report}
Report                      ::= CHOICE {
                                    [0] IMPLICIT DeliveredInfo,
                                    [1] IMPLICIT NonDeliveredInfo}
DeliveredInfo               ::= SET {
                                    delivery [0] IMPLICIT Time,
                                    typeOfUA [1] IMPLICIT INTEGER {
                                        public(0), private(1)}DEFAULT public}
NonDeliveredInfo            ::= SET {
                                    [0] IMPLICIT ReasonCode,
                                    [1] IMPLICIT DiagnosticCode OPTIONAL}
ReasonCode                  ::= INTEGER {
                                    transferFailure(0),
                                    unableToTransfer(1),
                                    conversionNotPerformed(2)}
```

FIGURE 17/X.411 (Part 1 of 2)

Formal definition of Delivery Report MPDU

```
-- P1 DEFINITIONS continued --
DiagnosticCode                  ::= INTEGER {
                                        unrecognizedORName(0),
                                        ambiguousORName(1),
                                        mtaCongestion(2),
                                        loopDetected(3),
                                        uaUnavailable(4),
                                        maximumTimeExpired(5),
                                        encodedInformationTypesUnsupported(6),
                                        contentTooLong(7),
                                        conversionImpractical(8),
                                        conversionProhibited(9),
                                        implicitConversionNotRegistered(10),
                                        invalidParameters(11)}
-- supplementary information
SupplementaryInformation        ::= PrintableString -- length limited and for further study --
```

FIGURE 17/X.411 (Part 2 of 2)

Formal definition of Delivery Report MPDU

```
-- P1 DEFINITIONS continued --
ProbeMPDU                       ::= ProbeEnvelope
ProbeEnvelope                   ::= SET {
                                        probe MPDUIdentifier,
                                        originator ORName,
                                        ContentType,
                                        UAContentID OPTIONAL,
                                        original EncodedInformationTypes OPTIONAL,
                                        TraceInformation,
                                        PerMessageFlag DEFAULT {},
                                        contentLength [0] IMPLICIT INTEGER OPTIONAL,
                                        [1] IMPLICIT SEQUENCE OF PerDomainBilateralInfo OPTIONAL,
                                        [2] IMPLICIT SEQUENCE OF RecipientInfo}
END -- of P1 DEFINITIONS --
```

FIGURE 18/X.411

Formal definition of Probe MPDU

```
P2 DEFINITIONS ::=
BEGIN

-- P2 makes use of types defined in the following modules:
--      P1: X.411, § 3.4
--      P3: X.411, § 4.3
--      SFD: this Recommendation, § 5
--      T73: T.73, § 5

UAPDU ::= CHOICE {
                [0] IMPLICIT IM-UAPDU,
                [1] IMPLICIT SR-UAPDU}

-- IP-message UAPDU

IM-UAPDU ::= SEQUENCE {Heading, Body}

-- heading

Heading ::= SET {
    IPMessageId,
    originator          [0] IMPLICIT ORDescriptor OPTIONAL,
    authorizingUsers    [1] IMPLICIT SEQUENCE OF ORDescriptor OPTIONAL,
                            -- only if not the originator
    primaryRecipients   [2] IMPLICIT SEQUENCE OF Recipient OPTIONAL,
    copyRecipients      [3] IMPLICIT SEQUENCE OF Recipient OPTIONAL,
    blindCopyRecipients [4] IMPLICIT SEQUENCE OF Recipient OPTIONAL,
    inReplyTo           [5] IMPLICIT IPMessageId OPTIONAL,
                            -- omitted if not in reply to a previous message
    obsoletes           [6] IMPLICIT SEQUENCE OF IPMessageId OPTIONAL,
    crossReferences     [7] IMPLICIT SEQUENCE OF IPMessageId OPTIONAL,
    subject             [8] CHOICE {T61String} OPTIONAL,
    expiryDate          [9] IMPLICIT Time OPTIONAL,
                            -- if omitted, expiry date is never
    replyBy             [10] IMPLICIT Time OPTIONAL,
    replyToUsers        [11] IMPLICIT SEQUENCE OF ORDescriptor OPTIONAL,
                            -- each O/R descriptor must contain an O/R name
    importance          [12] IMPLICIT INTEGER {low(0), normal(1), high(2)} DEFAULT normal,
    sensitivity         [13] IMPLICIT INTEGER {personal(1), private(2), companyConfidential(3)}
                             OPTIONAL,
    autoforwarded       [14] IMPLICIT BOOLEAN DEFAULT FALSE
      -- indicates that the forwarded message body part(s) were autoforwarded -- }

IP MessageId ::= [APPLICATION 11] IMPLICIT SET {
                        ORName OPTIONAL,
                        PrintableString}

ORName ::= P1.ORName

-- P2 definitions to be continued
```

FIGURE 3/X.420 (Part 1 of 3)

Formal definition of IM-UAPDU

```
-- P2 definitions continued

ORDescriptor        ::= SET {-- at least one of the first two members must be present
                        ORName OPTIONAL,
                        freeformName [0] IMPLICIT T61String OPTIONAL,
                        telephoneNumber [1] IMPLICIT PrintableString OPTIONAL}

Recipient           ::= SET {
                        [0] IMPLICIT ORDescriptor,
                        reportRequest [1] IMPLICIT BIT STRING {
                            receiptNotification(0),
                            nonReceiptNotification(1),
                            returnIPMessage(2)} DEFAULT {},
                            -- if requested, the O/R descriptor must contain an O/R name
                        replyRequest [2] IMPLICIT BOOLEAN DEFAULT FALSE
                            -- if true, the O/R descriptor must contain an O/R name --}

-- body

Body                ::= SEQUENCE OF BodyPart

BodyPart            ::= CHOICE {
                            [0] IMPLICIT IA5Text
                            [1] IMPLICIT TLX,
                            [2] IMPLICIT Voice,
                            [3] IMPLICIT G3Fax,
                            [4] IMPLICIT TIF0,
                            [5] IMPLICIT TTX,
                            [6] IMPLICIT Videotex,
                            [7] NationallyDefined,
                            [8] IMPLICIT Encrypted,
                            [9] IMPLICIT ForwardedIPMessage,
                           [10] IMPLICIT SFD,
                           [11] IMPLICIT TIF1}

-- body part types

IA5Text             ::= SEQUENCE {
                        SET {repertoire [0] IMPLICIT INTEGER {ia5(5), ita2(2)}
                            DEFAULTia5
                            -- additional members of this Set are a possible future extension --},
                        IA5String}

TLX                 ::= for further study

Voice               ::= SEQUENCE {
                        SET, -- members are for further study
                        BIT STRING}

-- P2 definitions to be continued
```

FIGURE 3/X.420 (Part 2 of 3)

Formal definition of IM-UAPDU

```
-- P2 definitions continued

G3Fax                    ::= SEQUENCE {
                                 SET {
                                     numberOfPages [0] IMPLICIT INTEGER OPTIONAL,
                                     [1] IMPLICIT P1.G3NonBasicParams OPTIONAL},
                                 SEQUENCE OF BIT STRING}

TIF0                     ::= T73Document

T73Document              ::= SEQUENCE OF T73.ProtocolElement

TTX                      ::= SEQUENCE {
                                 SET {
                                     numberOfPages [0] IMPLICIT INTEGER OPTIONAL,
                                     telexCompatible [1] IMPLICIT BOOLEAN DEFAULT FALSE,
                                     [2] IMPLICIT P1.TeletexNonBasicParams OPTIONAL},
                                 SEQUENCE OF T61String}

Videotex                 ::= SEQUENCE {
                                 SET, -- members are for further study
                                 VideotexString}

NationallyDefined        ::= ANY

Encrypted                ::= SEQUENCE {
                                 SET, -- members are for further study
                                 BIT STRING}

ForwardedIPMessage       ::= SEQUENCE {
                                 SET {
                                     delivery [0] IMPLICIT Time OPTIONAL,
                                     [1] IMPLICIT DeliveryInformation OPTIONAL},
                                 IM-UAPDU}

DeliveryInformation      ::= P3.DeliverEnvelope
                             -- This merely reuses a data type definition, and does not imply that the information
                             -- was ever carried in P3.

SFD                      ::= SFD.Document

-- note that SFD and T73 Document use the same space of application-wide tags which is different from that used for
-- other MHS protocols

TIF1                     ::= T73Document

-- P2 definitions to be continued
```

FIGURE 3/X.420 (Part 3 of 3)

Formal definition of IM-UAPDU

```
-- P2 definitions continued

-- IPM-status-report UAPDU

SR-UAPDU ::= SET {
        [0] CHOICE {
            nonReceipt  [0] IMPLICIT NonReceiptInformation,
            receipt     [1] IMPLICIT ReceiptInformation},
        reported            IPMessageId,
        actualRecipient     [1] IMPLICIT ORDescriptor OPTIONAL,
        intendedRecipient   [2] IMPLICIT ORDescriptor OPTIONAL,
            -- only present if not actual recipient the O/R descriptor must contain an
            -- O/R named
        converted           P1.EncodedInformationTypes OPTIONAL}

NonReceiptInformation ::= SET {
        reason [0] IMPLICIT
            INTEGER {uaeInitiatedDiscard(0), autoForwarded(1)},
        nonReceiptQualifier [1] IMPLICIT
            INTEGER {expired(0), obsoleted(1), subscriptionTerminated(2)}
            OPTIONAL,
        comments [2] IMPLICIT PrintableString OPTIONAL,
            -- on auto-forward
        returned [3] IMPLICIT IM-UAPDU OPTIONAL}

ReceiptInformation ::= SET {
        receipt [0] IMPLICIT Time,
        typeOfReceipt [1] IMPLICIT INTEGER {
            explicit(0),automatic(1)} DEFAULT explicit,
        [2] IMPLICIT P1.SupplementaryInformation OPTIONAL}

END -- of P2 definitions
```

FIGURE 4/X.420

Formal definition of SR-UAPDU

ANNEXE2 : LES CHAMPS D'UN RFC-822 MESSAGE ET LA LISTE ALPHABETIQUE DES REGLES DE LA SYNTAXE RFC-822

ALPHABETICAL LISTING OF SYNTAX RULES

```
address      =  mailbox                          ; one addressee
             /  group                            ; named list
addr-spec    =  local-part "@" domain            ; global address
ALPHA        =  <any ASCII alphabetic character>
                                                 ; (101-132, 65.- 90.)
                                                 ; (141-172, 97.-122.)
atom         =  1*<any CHAR except specials, SPACE and CTLs>
authentic    =   "From"        ":"   mailbox     ; Single author
             / ( "Sender"      ":"   mailbox     ; Actual submittor
                 "From"        ":" 1#mailbox)    ; Multiple authors
                                                 ;  or not sender
CHAR         =  <any ASCII character>            ; ( 0-177,  0.-127.)
comment      =  "(" *(ctext / quoted-pair / comment) ")"
CR           =  <ASCII CR, carriage return>      ; (     15,     13.)
CRLF         =  CR LF
ctext        =  <any CHAR excluding "(",         ; => may be folded
                 ")", "\" & CR, & including
                 linear-white-space>
CTL          =  <any ASCII control               ; (  0- 37,  0.- 31.)
                 character and DEL>              ; (    177,    127.)
date         =  1*2DIGIT month 2DIGIT            ; day month year
                                                 ;  e.g. 20 Jun 82
dates        =   orig-date                       ; Original
                 [ resent-date ]                 ; Forwarded
date-time    =  [ day "," ] date time            ; dd mm yy
                                                 ;  hh:mm:ss zzz
day          =  "Mon"  / "Tue" /  "Wed"  / "Thu"
             /  "Fri"  / "Sat" /  "Sun"
delimiters   =  specials / linear-white-space / comment
destination  =  "To"           ":" 1#address     ; Primary
             /  "Resent-To"    ":" 1#address
             /  "cc"           ":" 1#address     ; Secondary
             /  "Resent-cc"    ":" 1#address
             /  "bcc"          ":"  #address     ; Blind carbon
             /  "Resent-bcc"   ":"  #address
DIGIT        =  <any ASCII decimal digit>        ; ( 60- 71, 48.- 57.)
domain       =  sub-domain *("." sub-domain)
domain-literal =  "[" *(dtext / quoted-pair) "]"
domain-ref   =  atom                             ; symbolic reference
dtext        =  <any CHAR excluding "[",         ; => may be folded
                 "]", "\" & CR, & including
                 linear-white-space>
extension-field =
                <Any field which is defined in a document
                 published as a formal extension to this
                 specification; none will have names beginning
                 with the string "X-">
```

```
field       =  field-name ":" [ field-body ] CRLF
fields      =    dates                          ; Creation time,
                 source                         ;  author id & one
                1*destination                   ;  address required
                 *optional-field                ;  others optional
field-body  =  field-body-contents
               [CRLF LWSP-char field-body]
field-body-contents =
              <the ASCII characters making up the field-body, as
               defined in the following sections, and consisting
               of combinations of atom, quoted-string, and
               specials tokens, or else consisting of texts>
field-name  =  1*<any CHAR, excluding CTLs, SPACE, and ":">
group       =  phrase ":" [#mailbox] ";"
hour        =  2DIGIT ":" 2DIGIT [":" 2DIGIT]
                                                ; 00:00:00 - 23:59:59
HTAB        =  <ASCII HT, horizontal-tab>       ; (     11,       9.)
LF          =  <ASCII LF, linefeed>             ; (     12,      10.)
linear-white-space =  1*([CRLF] LWSP-char)      ; semantics = SPACE
                                                ; CRLF => folding
local-part  =  word *("." word)                 ; uninterpreted
                                                ; case-preserved
LWSP-char   =  SPACE / HTAB                     ; semantics = SPACE
mailbox     =  addr-spec                        ; simple address
            /  phrase route-addr                ; name & addr-spec
message     =  fields *( CRLF *text )           ; Everything after
                                                ;  first null line
                                                ;  is message body
month       =  "Jan"  /  "Feb" /  "Mar"  /  "Apr"
            /  "May"  /  "Jun" /  "Jul"  /  "Aug"
            /  "Sep"  /  "Oct" /  "Nov"  /  "Dec"
msg-id      =  "<" addr-spec ">"                ; Unique message id
optional-field =
            /  "Message-ID"        ":"   msg-id
            /  "Resent-Message-ID" ":"   msg-id
            /  "In-Reply-To"       ":"  *(phrase / msg-id)
            /  "References"        ":"  *(phrase / msg-id)
            /  "Keywords"          ":"  #phrase
            /  "Subject"           ":"  *text
            /  "Comments"          ":"  *text
            /  "Encrypted"         ":" 1#2word
            /  extension-field                  ; To be defined
            /  user-defined-field               ; May be pre-empted
orig-date   =  "Date"        ":"   date-time
originator  =   authentic                       ; authenticated addr
              [ "Reply-To"   ":" 1#address] )
phrase      =  1*word                           ; Sequence of words
```

```
qtext       =  <any CHAR excepting <">,      ; => may be folded
               "\" & CR, and including
               linear-white-space>
quoted-pair =  "\" CHAR                      ; may quote any char
quoted-string = <"> *(qtext/quoted-pair) <">; Regular qtext or
                                             ;   quoted chars.
received    =  "Received"    ":"             ; one per relay
                 ["from" domain]             ; sending host
                 ["by"   domain]             ; receiving host
                 ["via"  atom]               ; physical path
                *("with" atom)               ; link/mail protocol
                 ["id"   msg-id]             ; receiver msg id
                 ["for"  addr-spec]          ; initial form
                  ";"    date-time           ; time received

resent      =   resent-authentic
              [ "Resent-Reply-To"  ":" 1#address] )
resent-authentic =
            =   "Resent-From"      ":"   mailbox
            / ( "Resent-Sender"    ":"   mailbox
                "Resent-From"      ":" 1#mailbox  )
resent-date =  "Resent-Date" ":"   date-time
return      =  "Return-path" ":" route-addr ; return address
route       =  1#("@" domain) ":"           ; path-relative
route-addr  =  "<" [route] addr-spec ">"
source      = [  trace ]                    ; net traversals
               originator                    ; original mail
             [  resent ]                     ; forwarded
SPACE       =  <ASCII SP, space>            ; (     40,      32.)
specials    =  "(" / ")" / "<" / ">" / "@"  ; Must be in quoted-
            /  "," / ";" / ":" / "\" / <">  ;  string, to use
            /  "." / "[" / "]"              ;  within a word.
sub-domain  =  domain-ref / domain-literal
text        =  <any CHAR, including bare    ; => atoms, specials,
               CR & bare LF, but NOT        ;  comments and
               including CRLF>              ;  quoted-strings are
                                            ;  NOT recognized.
time        =  hour zone                    ; ANSI and Military
trace       =    return                     ; path to sender
               1*received                   ; receipt tags
user-defined-field =
              <Any field which has not been defined
               in this specification or published as an
               extension to this specification; names for
               such fields must be unique and may be
               pre-empted by published extensions>
word        =  atom / quoted-string
```

```
zone        =  "UT"  / "GMT"              ; Universal Time
                                          ; North American : UT
            /  "EST" / "EDT"              ;  Eastern:  - 5/ - 4
            /  "CST" / "CDT"              ;  Central:  - 6/ - 5
            /  "MST" / "MDT"              ;  Mountain: - 7/ - 6
            /  "PST" / "PDT"              ;  Pacific:  - 8/ - 7
            /  1ALPHA                     ; Military: Z = UT;
<">         =  <ASCII quote mark>         ; (     42,      34.)
```

ANNEXE 3 : LES CHAMPS D'EXTENSION UTILISES PAR LA PASSERELLE

```
extended-fields = "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)

                 / "TIFO"                         ; tifo (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
```

ANNEXE 4 : PROGRAMMES

```
/********************* PASSERELLE *****************/


#include <stdio.h>
#include "string.c"
#include "table.c"
#include "mapping.c"
#include "analyse.c"
#include "erreur.c"

#define maxadr 100

gt(adr,quoted,p1822,attribute,X400)
char adr[maxadr],attribute[maxadr],X400[maxadr];
int p1822,quoted;

{
int
ct2,passe,lg,ct,e1,e2,e3,e4,e5,posi,exist,er,e6,e7,e8,e9,e10,e11,e12,e13,e14;
char car,dom[maxadr],local[maxadr],table[maxadr],sortie[maxadr],domX400[maxadr
                                    ,ascii[maxadr],locX400[maxadr],prin[maxadr];
table[0] = 'p';
table[1] = 'a';
table[2] = 's';
table[3] = 's';
table[4] = 'e';
table[5] = '.';
table[6] = 'c';
table[7] = '\0';
ct2 = 1;
passe = 0;
lg = longueur(adr);
aadresse(adr,lg,&ct,&e1,&e2,&e3,&e4,&e5); /* etape 1.1 */
if (ct == 1)
  {
    car = '@';
    pos1 = dernier car(adr,car,lg);
    prendre(adr,pos1+1,lg,dom);
    prendre(adr,1,pos1-1,local);
    lg = lg - pos1;
    adomain(dom,lg,&ct,&e1,&e2,&e3,&e4);  /* etape 1.2. */
    if (e1 == 1)
       error(5);
    if (e2 == 1)
       error(6);
    if (e3 == 1)
       error(7);
    if (e4 == 1)
       error(8);
    if (ct == 1)
     {
       exist = chercher(table,dom,sortie);
       if (exist == 1)
            mapdomX400(dom,domX400);   /* etape 1.3. */
       else
         {
            creer(domX400);
            passe = 1;
         }
     }
    if (quoted == 1 || p1822 == 1)
         {
            maplocascii(local,ascii,&er);  /* etape 1.4. */
            copier(ascii,local);
         }
```

```
    lg= longueur(local);
    if (caractere(local,1) == '"' && caractere(local,lg) == '"')
        {
          prendre(local,2,lg-1,ascii);
          copier(ascii,local);
        }
    mapasciipr(local,prin,&er);     /* etape 1.5. */

    copier(prin,local);
    if ( er == 1)
        {
           error(9);
           ct2 = 0;
         }
    else
         {
           lg = longueur(local);
           astdorname(local,lg,&ct2,&e1,&e2,&e3,&e4,&e5); /* etape 1.6. */
           if (ct2 == 1)
                maporname(local,locX400);     /* etape 1.8 */
           else
             {
                aencodeapn(local,lg,&ct2,&e6,&e7,&e8,&e9); /* etape 1.7 */
                if (ct2 == 1 && e1 == 1 && e2 == 1)
                     mapencoded(local,locX400);  /* etape 1.8 */
                else
                  {
                     ct2 = 0;
                     if (e1 == 1)
                        error(10);
                     if (e2 == 1)
                        error(11);
                     if (e3 == 1)
                        error(12);
                     if (e4 == 1)
                        error(13);
                     if (e5 == 1)
                        error(14);
                     if (e6 == 1)
                        error(15);
                     if (e7 == 1)
                        error(16);
                     if (e8 == 1)
                        error(17);
                     if (e9 == 1)
                        error(18);
                  }
             }
         }
    if (ct == 1 && ct2 == 1)
         {
           concat(locX400,domX400);
           lg = longueur(locX400);
           aformat(locX400,X400,dom,passe,lg,&e1,&e2,&e3,&ct);
           if (e1 == 1)              /* etape 1.9. et 1.10. */
                error(19);
           if (e2 == 1)
                error(20);
           if (e3 == 1)
                error(21);
           if (ct == 0)
              crear(X400);
         }
  }
else if (e2 == 1)
```

```
    aroute(adr,lg,&ct,&e6,&e7,&e8,&e9,&e10,&e11,&e12,&e13);
    if (ct == 1)              /* etape 2.1. */
      {
         if (p1822 == 0)
          {
               mapasciipr(adr,prin,&er);    /* etape 2.3. */
               if ( er == 1)
                   error(9);
               else
                 {
                   concat(attribute,prin);  /* etape 2.4. */
                   dom[0] = 'T';
                   dom[1] = '\0';
                   exist = chercher(table,dom,sortie);
                   exist = chercher(table,sortie,dom);
                   if (exist == 1)
                         error(22);
                   else
                       {
                         concat(dom,attribute); /* etape 2.5. */
                         copier(dom,X400);
                       }
                 }
          }
         else
            error(31);            /* etape 2.2. */
      }
    else
      {
         if (e6 == 1)
            error(23);
         if (e7 == 1)
            error(24);
         if (e8 == 1)
            error(25);
         if (e9 == 1)
            error(26);
         if (e10 == 1)
            error(27);
         if (e11 == 1)
            error(28);
         if (e12 == 1)
            error(29);
         if (e13 == 1)
            error(30);
      }
 }
else
 {
    if (e1 == 1)
       error(1);
    if (e3 == 1)
       error(2);
    if (e4 == 1)
       error(3);
    if (e5 == 1)
       error(4);
 }
}
```

```
/****************** Module STRING ****************/


#include <stdio.h>
#define maxadr 100



/* Primitive1 : creer */
/* ------------------ */

creer(str)
char str[maxadr];
{
str[0] = '\0';
}


/* Primitive2 : longueur */
/* --------------------- */

longueur(str)
char str[maxadr];
{
int longu;
longu = strlen(str);
return(longu);
}


/* Primitive3 : caractere */
/* ---------------------- */

caractere(str,position)
char str[maxadr];
int  position;
{
 char car;
car = str[position-1];
return(car);
}


/* Primitive4 : ajout */
/* ------------------ */

ajout(str,car,longu)
char str[maxadr], car;
int longu;
{
str[longu] = car;
str[longu+1] = '\0';
}
```

```
/* Primitive5 : cherchercar */
/* ------------------------ */

cherchercar(str,pos1,car)
char str[maxadr], car;
int pos1;
{
int pos2;
while ( str[pos1] != car && str[pos1] != '\0' )
    pos1++;
if (str[pos1] == car)
    pos2 = pos1+1;
else
    pos2 = 0;
return(pos2);
}


/* Primitive6 : derniercar */
/* ----------------------- */

derniercar(str,car,longu)
char str[maxadr], car;
int longu;
{
int pos;
pos = longu-1;
while ( pos != 0 && str[pos] != car)
   pos--;
if ( str[pos] == car)
   pos = pos+1;
else
   pos = 0;
return(pos);
}


/* Primitive7 : nbrecar */
/* -------------------- */

nbrecar(str,car,pos1,pos2)
char str[maxadr], car;
int pos1, pos2;
{
int nbre;
nbre = 0;
pos1 = pos1-1;
while (pos1 != pos2)
  {
   if (str[pos1] == car)
     nbre++;
   pos1++;
  }
return(nbre);
}
```

```
/* Primitive8 : prendre */
/* ------------------- */

prendre(str1,pos1,pos2,str2)
char str1[maxadr], str2[maxadr];
int pos1,pos2;
{
int i;
i = 0;
pos1 = pos1-1;
while (pos1 != pos2)
  {
   str2[i] = str1[pos1];
   i++;
   pos1++;
  }
str2[i] = '\0';
}


/* Primitive9 : concat */
/* ------------------- */
concat(str1,str2)
char str1[maxadr],str2[maxadr];
{
strcat(str1,str2);
}


/* Primitive10 : copier */
/* -------------------- */

copier(str1,str2)
char str1[maxadr],str2[maxadr];
{
strcpy(str2,str1);
}


/* Primitive11 : replace */
/* --------------------- */

replace(str1,car1,car2)
char str1[maxadr], car1,car2;
{
int j;
j=0;
while (str1[j] != '\0')
  {
   if (str1[j] == car1)
      str1[j] = car2;
   j++;
  }
}
```

```
/* Primitive12 : comparer */
/* ---------------------- */

comparer(str1,str2)
char str1[maxadr],str2[maxadr];
{
int egal;
egal = strcmp(str1,str2);
return(egal);
}


/* Primitive13 : inserer */
/* --------------------- */

inserer(str1,str2,pos)
char str1[maxadr],str2[maxadr];
int pos;
{
int n;
char part[maxadr],part2[maxadr];
n = longueur(str1);
if (pos > 1)
  prendre(str1,1,pos-1,part);
else
  creer(part);
concat(part,str2);
if (pos < n+1)
  {
   prendre(str1,pos,n,part2);
   concat(part,part2);
  }
copier(part,str1);
}


/* Primitive14 : changer */
/* --------------------- */

changer(str,pos,car)
char str[maxadr],car;
int pos;
{
str[pos-1] = car;
}
```

```
/*****************  Module TABLE ***************/


/* Primitive1 : liretable */
/* ---------------------- */

liretable(s,ftable)

char s[maxadr];
FILE *ftable;
{
int c,i,lim;
i=0;
lim = maxadr;
while (--lim > 0 && (c=getc(ftable)) != EOF && c != '\n')
  s[i++] =c;
if (c == '\n')
  s[i++] = '\0';
if (c == EOF)
  return(-1);
s[i] = '\0';
return(i);
}


/* Primitive2 : chercher */
/* --------------------- */

chercher(table,entree,sortie)
char table[maxadr],entree[maxadr],sortie[maxadr];
{
FILE  *ftable;
char line[maxadr],init[maxadr],che[maxadr];
int pos1,pos2,egal,stop,n;
che[0] = '/';
che[1] = 'u';
che[2] = 's';
che[3] = 'e';
che[4] = 'r';
che[5] = 's';
che[6] = '/';
che[7] = 's';
che[8] = 't';
che[9] = 'u';
che[10] = 'd';
che[11] = 'e';
che[12] = 'n';
che[13] = 't';
che[14] = 's';
che[15] = '/';
che[16] = 'v';
che[17] = 'f';
che[18] = 'a';
che[19] = '/';
che[20] = '\0';
```

```
stop =1;
egal = 0;
concat(che,table);
ftable = fopen(che,"r");
while ((liretable(line,ftable) > 0) && stop != 0)
  {
   posi = cherchercar(line,0,' ');
   if (posi != 1)
     prendre(line,1,posi-1,init);
   else
     creer(init);
   egal = comparer(entree,init);
   if (egal == 0)
     {
       n = longueur(line);
       pos2 = cherchercar(line,posi,'*');
       prendre(line,pos2+1,n,sortie);
       stop = 0;
     }
   else
   egal = 1;
  }
return(egal);
}
```

```
/******************** Module ANALYSE ****************/


/* Primitive1 : aadresse */
/* --------------------- */

aadresse(adresse,longu,correct,erreur1,erreur2,erreur3,erreur4,erreur5)
int  longu,*correct,*erreur1,*erreur2,*erreur3,*erreur4,*erreur5;
char adresse[maxadr];

{
int  pos4,pos1,pos2,pos3,intaux;
char caractaux;

*correct = 1;
*erreur1 = 0;
*erreur2 = 0;
*erreur3 = 0;
*erreur4 = 0;
*erreur5 = 0;
caractaux = '"';

pos4 = dernier car(adresse,caractaux,longu);
caractaux = '@';
pos1 = dernier car(adresse,caractaux,longu);
intaux = 0;
pos2 = cherchercar(adresse,intaux,caractaux);
pos3 = cherchercar(adresse,pos4,caractaux);
if (pos2 == 0)
   {
    *erreur1 = 1;  /* pas de caractere "@"*/
    *correct = 0;
   }
if (pos2 == 1)
   {
    *erreur2 = 1;  /* "@" en premiere position */
    *correct = 0;
   }
if (pos1 == longu)
   {
    *erreur3 = 1;  /* "@" en derniere position */
    *correct = 0;
   }
if (pos1 < pos4)
   {
    *erreur4 = 1;  /* "@" suivi d'un caractere <"> */
    *correct = 0;
   }
if (pos3 != pos1)
   {
    *erreur5 = 1;  /* plusieurs "@" apres le dernier <"> */
    *correct = 0;
   }

}
```

```
/* Primitive2 : adomain */
/* -------------------- */

adomain(domaine,longu,correct,erreur1,erreur2,erreur3,erreur4)

int  longu,*correct,*erreur1,*erreur2,*erreur3,*erreur4;
char domaine[maxadr];

{
int  iaux,i;
char car;

i = 1;
*correct =1;
*erreur1 = 0;
*erreur2 = 0;
*erreur3 = 0;
*erreur4 = 0;
car = caractere(domaine,i);
if ((car<'0' && car!='.' && car!='-') || (car>'9' && car<'A') || (car>'Z'
                                                  && car<'a') || (car>'z'))
   {
     *erreur1 = 1;  /* caractere illegal */
     *correct = 0;
   }

if ((car<'A') || (car>'Z' && car<'a') || (car>'z'))
   {
     *erreur2 = 1;   /* premier caractere different d'une lettre */
     *correct = 0;
   }

while (i<longu-1)
   {
     i++;
     car = caractere(domaine,i);
     if ((car<'0' && car!='.' && car!='-') || (car>'9' && car<'A') || (car>'Z'
                                                  && car<'a') || (car>'z'))
        {
          *erreur1 = 1;   /* caractere illegal */
          *correct = 0;
        }

     if (car =='.')
        {
          iaux =i-1;
          car = caractere(domaine,iaux);
          if (car == '.')
              {
                *erreur4 = 1;  /* domaine vide */
                *correct = 0;
              }
          else if ((car<'0') || (car>'9' && car<'A') || (car>'Z' && car<'a')
              {                                                 || (car>'z'))
                *erreur2 = 1;  /* premier caractere apres le point different */
                *correct = 0;  /* d'une lettre */
              }
          iaux=i+1;
          car = caractere(domaine,iaux);
          if ((car<'0' && car!='.' && car!='-') || (car>'9' &&car<'A') ||
              {                          (car>'Z' && car<'a') || (car>'z'))
                *erreur1 = 1;    /* caractere illegal */
                *correct = 0;
              }
```

```
        if (car == '.')
            {
              *erreur4 = 1;      /* domaine vide */
              *correct = 0;
            }
        else if ((car<'A') || (car>'Z' && car<'a') || (car>'z'))
            {
              *erreur3 = 1;      /* dernier caractere different d'une */
              *correct = 0;      /* lettre ou d'un chiffre */
            }
        i++;
      }
   }
car = caractere(domaine,longu);
if ((car<'0' && car!='.' && car!='-') || (car>'9' && car<'A') || (car>'Z' &&
   {                                                          car<'a') || (car>'z'))
      *erreur1 = 1;     /* caractere illegal */
      *correct = 0;
   }
if ((car<'0') || (car>'9' && car<'A') || (car>'Z' && car<'a') || (car>'z'))
   {
      *erreur3 = 1;              /* dernier caractere different d'une */
      *correct = 0;              /* lettre ou d'un chiffre */
   }
}


/* Primitive3 : aencodedpn */
/* ----------------------- */

aencodedpn(encodedpn,longu,correct,erreur1,erreur2,erreur3,erreur4)

int  longu,*correct,*erreur1,*erreur2,*erreur3,*erreur4;
char encodedpn[maxadr];

{
int stop,first,pos1,nbre,pos2,i;
char car;

stop = 0;
*correct = 1;
*erreur1 = 0;
*erreur2 = 0;
*erreur3 = 0;
*erreur4 = 0;
first = 1;
pos1 = 0;
car = '.';
nbre = nbrecar(encodedpn,car,1,longu);
while (nbre != 0 && stop == 0)
  {
    car = '.';
    pos2 = cherchercar(encodedpn,pos1,car);
    switch (pos2-pos1)  {
     case 1 : *erreur3 = 1;  /* surname incorrect */
              *correct = 0;
              break;
     case 2 : car = caractere(encodedpn,pos2-1);
              if ((car<'A') || (car>'Z' && car<'a') || (car>'z'))
                      {
                         *erreur1 = 1;  /* initial incorrect */
                         *correct = 0;
                      }
              break;
```

```
      default : if (first == 1)
                  {
                    i = pos1+1;
                    while (i != pos2)
                      {
                        car = caractere(encodedpn,i);
                        if (car == '(')
                            {
                              *erreur2 = 1; /* surname incorrect */
                              *correct = 0;
                            }
                         i++;
                      }
                  }
                else
                     stop = 1;
                break;
                        }
     if (stop == 0)
               {
                 pos1= pos2;
                 nbre = nbre-1;
                 first = 0;
               }
   }
if (stop == 0 && pos2 == longu && *erreur3 == 0)
    {
         *erreur4 = 1;     /* pas de surname */
         *correct = 0;
    }
}


/* Primitive4 : alocalatom */
/* ----------------------- */

alocalatom(localpart,longu,correct,erreur1,erreur2,erreur3,erreur4)

int  longu,*correct,*erreur1,*erreur2,*erreur3,*erreur4;
char localpart[maxadr];


{
int i,n,pos;
char car;
char special[maxadr];
special[0] = '\\';
special[1] = ' ';
special[2] = '(';
special[3] = ')';
special[4] = '<';
special[5] = '>';
special[6] = '@';
special[7] = ';';
special[8] = ':';
special[9] = '"';
special[10] = ',';
special[11] = '[';
special[12] = ']';
special[13] = '\0';
*erreur1 = 0;
*erreur2 = 0;
*erreur3 = 0;
*erreur4 = 0;
*correct = 1;
```

```
i = 1;
car = caractere(localpart,i);
pos = cherchercar(special,0,car);
if ((pos != 0) || (car < ' '))
    {
      *erreur1 = 1;   /* caractere illegal */
      *correct = 0;
    }
if (car == '.')
    {
      *erreur3 = i;   /* premier caractere est un point */
      *correct = 0;
    }
while (i <= longu-1)
    {
      i=i+1;
      car = caractere(localpart,i);
      pos = cherchercar(special,0,car);
      if ((pos != 0) || (car < ' '))
        {
           *erreur1 = 1; /* caractere illegal */
           *correct = 0;
        }
      if (car == '.')
        {
           car = caractere(localpart,i-1);
           if (car == '.')
             {
               *erreur4 = 1;  /* atom vide */
               *correct = 0;
             }
        }
    }
car = caractere(localpart,longu);
if (car == '.')
    {
       *erreur2 = 1;   /* un point termine la localpart */
       *correct = 0;
    }
}


/* Primitive5 : astdorname */
/* ----------------------- */

astdorname(stdorname,longu,correct,erreur1,erreur2,erreur3,erreur4,erreur5)

int  longu,*correct,*erreur1,*erreur2,*erreur3,*erreur4,*erreur5;
char stdorname[maxadr];

{
int pos1,nbre,pos2,egal,pos3,nbre1,lg,e1,e2,e3,e4,ct;
char quoted[maxadr],inter[maxadr],table[maxadr],sortie[maxadr],car,car1,aux;
table[0] ='a';
table[1] = 't';
table[2] ='t';
table[3] = 'r';
table[4] = 'i';
table[5] = 'b';
table[6] = 'u';
table[7] = 't';
table[8] = 'e';
table[9] = '.';
table[10] = 'c';
table[11] = '\0';
```

```
 pos1 = 0;
 aux = "/";
 pos2 = 1;
 nbre = nbrecar(stdorname,aux,pos2,longu);
 while (nbre != 0)
   {
     aux = "/";
     pos2 = chercnercar(stdorname,pos1,aux);
     if (pos2 > 5)
        {
          inter[0] = "(";
          inter[1] = "0";
          inter[2] = "3";
          inter[3] = "6";
          inter[4] = ")";
          inter[5] = "\0";
          prendre(stdorname,pos2-5,pos2-1,quoted);
          egal = comparer(quoted,inter);
          if (egal == 0)
             {
               aux = "$";
               changer(stdorname,pos2,aux);
             }
        }
     nbre = nbre-1;
     pos1 = pos2;
   }
 pos1 = 0;
 aux = "=";
 nbre = nbrecar(stdorname,aux,1,longu);
 while (nbre != 0)
   {
     aux = "=";
     pos2 = chercnercar(stdorname,pos1,aux);
     if (pos2 > 5)
        {
          inter[0] = "(";
          inter[1] = "0";
          inter[2] = "3";
          inter[3] = "6";
          inter[4] = ")";
          inter[5] = "\0";
          prendre(stdorname,pos2-5,pos2-1,quoted);
          egal = comparer(quoted,inter);
          if (egal == 0)
             {
               aux = "%";
               changer(stdorname,pos2,aux);
             }
        }
     nbre = nbre-1;
     pos1 = pos2;
   }
 *correct = 1;
 *erreur1 = 0;
 *erreur2 = 0;
 *erreur3 = 0;
 *erreur4 = 0;
 *erreur5 = 0;
 pos1 = 1;
 car = caractere(stdorname,pos1);
 if (car != "/")
   {
     *erreur1 = 1;  /* premier caractere different de "/" */
     *correct = 0;
   }
```

```
car = caractere(stdorname,longu);
if (car != '/')
  {
    *erreur2 = 1;  /* dernier caractere different de "/" */
    *correct = 0;
  }
car = '/';
nbre = nbrecar(stdorname,car,pos1,longu);
if (nbre > 1)
   {
     pos1 = 0;
     pos1 = cherchercar(stdorname,pos1,car);
     nbre = nbre-1;
     while (nbre != 0)
          {
            car = '/';
            pos2 = cherchercar(stdorname,pos1,car);
            car = '=';
            nbre1 = nbrecar(stdorname,car,pos1,pos2);
                if (nbre1 != 1)
                   {
                    *erreur3 = 1;   /* zero ou plusieurs caracteres "=" */
                    *correct = 0;
                   }
                if (nbre1 >= 1)
                   {
                     car = '=';
                     pos3 = cherchercar(stdorname,pos1,car);
                     if (pos3 - pos1 > 1)
                          {
                            prendre(stdorname,pos1+1,pos3-1,inter);
                            egal = chercher(table,inter,sortie);
                            if (egal == 1)
                                 {
                                     *erreur4 = 1; /* attribut errone */
                                     *correct = 0;
                                 }
                          }
                   }
            nbre = nbre-1;
            pos1 = pos2;
          }
   }
if (*correct == 1)
   {
      pos1 = 0;
      do {
           car = 'P';
           pos1 = cherchercar(stdorname,pos1,car);
           if ( longu - pos1 > 3)
             {
               car = caractere(stdorname,pos1+1);
               car1 = caractere(stdorname,pos1+2);
               if (car == 'N' && car1 == '=')
                 {
                  car = '/';
                  pos2 = cherchercar(stdorname,pos1+2,car);
                  prendre(stdorname,pos1+3,pos2-1,quoted);
                  lg = longueur(quoted);
                  decodedpn(quoted,lg,&ct,&e1,&e2,&e3,&e4);
                  if (ct == 0)
                     {
                       *erreur5 = 1; /* "PN" incorrect */
                       *correct = 0;
                     }
                 }
             }
        } while ( pos1 != 0);
```

```
car = '$';
car1 = '/';
replace(stdorname,car,car1);
car = '%';
car1 = '=';
replace(stdorname,car,car1);
}


/* Primitive6 : alocalword */
/* ----------------------- */

alocalword(localpart,longu,correct,erreur1,erreur2,erreur3)

int  longu,*correct,*erreur1,*erreur2,*erreur3;
char localpart[maxadr];

{
int pos1,quoted,pos2,pos;
char car,special[maxadr];

special[0] = '\\';
special[1] = ' ';
special[2] = '(';
special[3] = ')';
special[4] = '<';
special[5] = '>';
special[6] = '@';
special[7] = ';';
special[8] = ':';
special[9] = '"';
special[10] = ',';
special[11] = '[';
special[12] = ']';
special[13] = '\0';
*correct = 1;
*erreur1 = 0;
*erreur2 = 0;
*erreur3 = 0;
pos1 = 1;
while (pos1 != longu+1)
  {
    car = caractere(localpart,pos1);
    switch (car) {
    case '.' : *erreur1 = 1; /* word non present */
               *correct = 0;
               pos1 = pos1+1;
               break;
    case '"' : quoted = 0;
               while (pos1 != 0 && quoted == 0)
                 {
                   car = '"';
                   pos2 = cherchercar(localpart,pos1,car);
                   if (pos2 > 1)
                       {
                         car = caractere(localpart,pos2-1);
                         if (car != '\\')
                             quoted = 1;
                       }
                   pos1 = pos2;
                 }
               if (pos1 == 0)
                 {
                   *erreur3 = 1;  /* quoted-string non borne */
                   *correct = 0;
                   pos1 = longu ;
                 }
```

```
                else
                  {
                    if (pos1 != longu)
                       {
                          pos1 = pos1 +1;
                          car = caractere(localpart,pos1);
                          if (car != '.')
                             {
                                *erreur2 = 1;  /* syntaxe d'atom incorrecte */
                                *correct = 0;
                                car = '.';
                                pos2 = cherchercar(localpart,pos1,car);
                                if (pos2 == 0)
                                     pos1 = longu;
                             }
                       }
                  }
                pos1 = pos1 + 1;
                break;
    default   : car = '.';
                pos2 = cherchercar(localpart,pos1,car);
                if (pos2 == 0)
                   pos2 = longu;
                while (pos1 != pos2)
                   {
                      car = caractere(localpart,pos1);
                      pos = cherchercar(special,0,car);
                      if ((pos != 0) || (car < ' '))
                         {
                           *erreur2 = 1;  /* syntaxe d'atom incorrecte */
                           *correct = 0;
                         }
                           pos1 = pos1 +1;
                   }
                pos1 = pos1 +1;
                break;
                  }
  }
car = caractere(localpart,longu);
if (car == '.')
   {
     *erreur1 = 1;
     *correct = 0;
   }
}

/* Primitive7 : aroute */
/* ------------------- */


aroute(route,longu,correct,erreur1,erreur2,erreur3,erreur4,erreur5,erreur6,
                                                          erreur7,erreur8)
char route[maxadr];
int longu,*correct,*erreur1,*erreur2,*erreur3,*erreur4,*erreur5,*erreur6,
                                                  *erreur7,*erreur8;
{
int pos1,lg,ct,e1,e2,e3,e4,e5,pos2,pos,pos3;
char car,adrspec[maxadr],domaine[maxadr],local[maxadr];

*correct = 1;
*erreur1 = 0;
*erreur2 = 0;
*erreur3 = 0;
*erreur4 = 0;
*erreur5 = 0;
*erreur6 = 0;
*erreur7 = 0;
*erreur8 = 0;
```

```
pos1 = 0;
car = ':';
pos1 = cherchercar(route,pos1,car);
if (pos1 == 0)
   {
     *erreur1 = 1; /* pas de caractere ":" */
     *correct = 0;
   }
else
   {
     if (pos1 == longu)
         {
          *erreur2 = 1;    /* absence de l'adresse specifique */
          *correct = 0;
         }
     else
         {
          prendre(route,pos1+1,longu,adrspec);
          lg = longu - pos1;
          aadresse(adrspec,lg,&ct,&e1,&e2,&e3,&e4,&e5);
          if (ct == 0)
              {
                *erreur3 = 1;  /* adresse specifique erronee */
                *correct = 0;
              }
          else
              {
                car = '@';
                pos2 = derniercar(adrspec,car,lg);
                prendre(adrspec,pos2+1,lg,domaine);
                prendre(adrspec,1,pos2-1,local);
                lg = lg-pos2;
                adomain(domaine,lg,&ct,&e1,&e2,&e3,&e4);
                if (ct == 0)
                   {
                     *erreur4 = 1; /* domaine errone */
                     *correct = 0;
                   }
                lg = pos2-1;
                alocalword(local,lg,&ct,&e1,&e2,&e3);
                if (ct == 0)
                   {
                     *erreur5 = 1; /* local-part erronee */
                     *correct = 0;
                   }
              }
         }
     if (pos1 == 2)
         {
            *erreur6 = 1;  /* pas de domaine dans route */
            *correct = 0;
         }
      else
         {
            prendre(route,1,pos1-1,local);
            car = '@';
            pos2 = derniercar(local,car,pos1-1);
            if ( pos1-pos2 == 1)
               {
                 *erreur6 = 1;   /* absence d'un domaine dans route */
                 *correct = 0;
               }
            else
               {
                 prendre(local,pos2+1,pos1-1,domaine);
                 lg = pos1-1-pos2;
                 adomain(domaine,lg,&ct,&e1,&e2,&e3,&e4);
                 if (ct == 0)
```

```
                    {
                      *erreur7 = 1;  /* domaine errone */
                      *correct = 0;
                    }
                }
              while (pos2 !=1)
                {
                  pos2 = pos2-1;
                  car = caractere(local,pos2);
                  pos = 0;
                  while (car == ',')
                    {
                      pos = pos2-1;
                      car = caractere(local,pos);
                    }
                  if (pos == 0)
                    {
                      *erreur8 = 1;  /* manque une virgule */
                      *correct = 0;
                    }
                  else
                      pos2 = pos ;
                  car = '@';
                  pos3 = dernier car(local,car,pos2);
                  if (pos2-pos3 == 1)
                    {
                      *erreur6 = 1; /* absence d'un domaine */
                      *correct = 0;
                    }
                  else
                    {
                      prendre(local,pos3+1,pos2,domaine);
                      lg  = pos2-pos3;
                      adomain(domaine,lg,&ct,&e1,&e2,&e3,&e4);
                      if (ct == 0)
                          {
                             *erreur7 = 1; /* domaine errone */
                             *correct = 0;
                          }
                    }
                  pos2 = pos3;
                }
          }
    }
}


/* Primitive8 : aattribut */
/* ---------------------- */

aattribut(sortie,c,a,p,o)
int *c,*a,*p,*o;
char sortie[maxadr];

{
int pos1;
char car;
*c = 0;
*a = 0;
*p = 0;
*o = 0;
pos1 = 0;
car = '=';
```

```
posl = chercnercar(sortie,posl,car);
while (posl != 0)
  {
    car = caractere(sortie,posl-1);
    switch (car) {
    case 'C' : *c = 1;
               break;
    case 'A' : *a = 1;
               break;
    case 'P' : *p = 1;
               break;
    case 'O' : *o = 1;
               break;
    default : break;
                }
    car = '=';
    posl = chercnercar(sortie,posl,car);
  }
}


/* primitive9 : aformat */
/* -------------------- */

aformat(adresse,X400,passerelle,pas,longu,erreur1,erreur2,erreur3,correct)

char adresse[maxadr],X400[maxadr],passerelle[maxadr];
int longu,*erreur1,*correct,*erreur3,pas,*erreur2;

{
int pos1,pos2,c,a,uucp,p,x121,type,tid,uaid,o,u,s,gi,i,gq,rfc822,mail;
char car,car1,valeur[maxadr],table[maxadr];

table[0] = 'p';
table[1] = 'a';
table[2] = 's';
table[3] = 's';
table[4] = 'e';
table[5] = '.';
table[6] = 'c';
table[7] = '\0';
c = 0;
a = 0;
uucp = 0;
p = 0;
x121 = 0;
tid = 0;
uaid = 0;
o = 0
;
u = 0;
s = 0;
gi = 0;
i = 0;
gq = 0;
rfc822 = 0;
mail = 0;
*correct = 1;
*erreur1 = 0;
*erreur2 = 1;
*erreur3 = 0;
pos1 = 1;
car = '=';
pos1 = cherchercar(adresse,pos1,car);
```

```
while (pos1 != 0)
  {
    car = caractere(adresse,pos1-1);
    car1 = ";";
    pos2 = cherchercar(adresse,pos1,car1);
    if (pos2 == 0)
       pos2 = longu + 1;
    prendre(adresse,pos1+1,pos2-1,valeur);
    switch (car) {
    case "C" : if (c == 1)
                 {
                   *erreur1 = 1;  /* deux attributs identiques */
                   *correct = 0;
                 }
               else
                   c = 1;
               break;
    case "A" : if (a == 1)
                 {
                   *erreur1 = 1;  /* deux attributs identiques */
                   *correct = 0;
                 }
               else
                   a = 1;
               break;
    case "P" : if (pos1 > 2)
                   car = caractere(adresse,pos1-2);
               else
                   car = "%";
               if (car == "C")
                 {
                   if (uucp == 1)
                       {
                         *erreur1 = 1; /* deux attributs identiques */
                         *correct = 0;
                       }
                   else
                         uucp = 1;
                 }
               else
                 {
                   if (p == 1)
                       {
                         *erreur1 = 1; /* deux attributs identiques */
                         *correct = 0;
                       }
                   else
                         p = 1;
                 }
               break;
    case "1" : if (x121 == 1)
                 {
                   *erreur1 = 1; /* deux attributs identiques */
                   *correct = 0;
                 }
               else
                 {
                   x121 = 1;
                   type = numeric(valeur);
                   if (type == 0)
                       {
                         *erreur3 = 1; /* valeur incorrecte */
                         *correct = 0;
                       }
                 }
               break;
```

```
case 'D' : car = caractere(adresse,posi-4);
           if (car == 'T')
             {
               if (tid == 1)
                   {
                     *erreur1 = 1;  /* deux attributs identiques */
                     *correct = 0;
                   }
               else
                   {
                     tid = 1;
                     type = numeric(valeur);
                     if (type == 0)
                        {
                            *erreur3 = 1; /* valeur incorrecte */
                            *correct = 0;
                        }
                   }
             }
           else
             {
               if (uaid == 1)
                   {
                     *erreur1 = 1;  /* deux attributs identiques */
                     *correct = 0;
                   }
               else
                   {
                     uaid = 1;
                     type = numeric(valeur);
                     if (type == 0)
                        {
                            *erreur3 = 1; /* valeur incorrecte */
                            *correct = 0;
                         }
                   }
             }
           break;
case 'O' : if (o == 1)
             {
               *erreur1 = 1; /* deux attributs identiques */
               *correct = 0;
             }
           else
               o = 1;
           break;
case 'U' : u = 1;
           break;
case 'S' : if (s == 1)
             {
               *erreur1 = 1;  /* deux attributs identiques */
               *correct = 0;
             }
           else
               s = 1;
           break;
case 'I' : if (posi > 2)
               car = caractere(adresse,posi-2);
           else
               car = '%';
           if (car == 'G')
             {
               if (gi == 1)
                   {
                     *erreur1 = 1;  /* deux attributs identiques */
                     *correct = 0;
                   }
```

```
                    else
                          g1 = 1;
                  }
                else
                  {
                    if (i == 1)
                        {
                          *erreur1 = 1; /* deux attributs identiques */
                          *correct = 0;
                        }
                    else
                          i = 1;
                  }
                break;
    case 'Q' : if (gq == 1)
                  {
                    *erreur1 = 1;    /* deux attributs identiques */
                    *correct = 0;
                  }
                else
                    gq = 1;
                break;
    case '2' : if (rfc822 == 1)
                  {
                    *erreur1 = 1;  /* deux attributs identiques */
                    *correct = 0;
                  }
                else
                    rfc822 = 1;
                break;
    case 'L' : if (mail == 1)
                  {
                    *erreur1 = 1;  /* deux attributs identiques */
                    *correct = 0;
                  }
                else
                    mail = 1;
                break;
     default : break;
                  }
     car = '=';
     posi = cherchercar(adresse,posi,car);
  }
if ( c == 0 && a == 0 && pas == 1)
     {
       pas = chercher(table,passerelle,X400);
       concat(X400,adresse);
       c = 1;
       a = 1;
     }
else
```

```
        copier(adresse,X400);
pas = c*a;
pos1 = p + s + o + u + rfc822 + mail + uucp;
pos2 = s + gi + gq + i;
type = x121 + uaid + tid;
if ( pas == 1 && pos1 > 0 && ( pos2 == 0 || s == 1 ) && type == 0 )
   *erreur2 = 0;      /* forme 1 et variante 1 d'un ORName */
pas = c * a * uaid;
pos1 = p + s + gi + i + gq + o + u + x121 + tid;
if ( pas == 1 && pos1 == 0 )
   *erreur2 = 0;      /* forme 1 et variante 2 d'un ORName */
pas = c * a * x121;
pos1 = p + s + gi + i + gq + o + u + uaid + tid;
if ( pas == 1 && pos1 == 0 )
   *erreur2 = 0;      /* forme 1 et variante 3 d'un ORName */
pos1 = p + s + gi + i + gq + o + u + uaid + rfc822 + mail + uucp + c + a;
if ( x121 == 1 && pos1 == 0)
   *erreur2 = 0;      /* forme 2 d'un ORName */
if (*erreur2 == 1)
   *correct = 0;
}


numeric(valeur)

char valeur[maxadr];

{
int lg,i,exist;
char car;

lg = longueur(valeur);
i = 0;
exist = 1;
while (i != lg && exist == 1)
  {
    i = i + 1;
    car = caractere(valeur,i);
    if ( car < '0' || car > '9')
         exist = 0;
  }
return(exist);
}
```

```
/************************** Module MAPPING ********************/


/* Primitive1 mapasciipr */
/* --------------------- */

mapasciipr(ascii,printable,erreur)
char ascii[maxadr],printable[maxadr];
int *erreur;
{
int lg,longu,i,exist;
char car,partie[maxadr],table[maxadr],sortie[maxadr];
table[0] = 'a';
table[1] = 's';
table[2] = 'c';
table[3] = 'i';
table[4] = 'i';
table[5] = 'p';
table[6] = 'r';
table[7] = '.';
table[8] = 'c';
table[9] = '\0';
longu = longueur(ascii);
*erreur = 0;
creer(printable);
i = 0;
while (i != longu  && *erreur == 0)
  {
    i = i+1;
    car = caractere(ascii,i);
    if ((car < '\"' && car != ' ') || (car > ':' && car < 'A' && car != '=' &&
car != '?') || (car == '*') || (car > 'Z' && car < 'a') || (car > 'z'))
      {
         prendre(ascii,i,i,partie);
         exist = chercher(table,partie,sortie);
         if (exist == 0)
               concat(printable,sortie);
         else
               *erreur = 1;
      }
    else
      {
         lg = longueur(printable);
         ajout(printable,car,lg);
      }
  }
if (*erreur == 1)
  {
    creer(printable);
    copier(ascii,printable);
  }
}
```

```
/* Primitive2 maplocascii */
/* ---------------------- */


maplocascii(localpart,ascii,erreur)
char localpart[maxadr],ascii[maxadr];
int *erreur;

{
int e1,e2,e3,e4,ct,lg,nbre,pos1,pos2,longu,exist;
char car,partie[maxadr],table[maxadr],sortie[maxadr],car1;

table[0] = 'l';
table[1] = 'o';
table[2] = 'c';
table[3] = 'a';
table[4] = 's';
table[5] = 'c';
table[6] = 'i';
table[7] = 'i';
table[8] = '.';
table[9] = 'c';
table[10] = '\0';
*erreur = 0;
creer(ascii);
longu = longueur(localpart);
alocalatom(localpart,longu,&ct,&e1,&e2,&e3,&e4);
if ( ct == 1)
   {
     car = '#';
     nbre = nbrecar(localpart,car,1,longu);
     if (nbre % 2  == 1)
          ct = 0;
     else
        {
          pos1 = 0;
          while (nbre != 0 && *erreur == 0)
           {
                car = '#';
                pos2 = cherchercar(localpart,pos1,car);
                if (pos2 - pos1 > 1)
                      {
                        prendre(localpart,pos1+1,pos2-1,partie);
                        concat(ascii,partie);
                      }
                pos1 = cherchercar(localpart,pos2,car);
                prendre(localpart,pos2,pos1,partie);
                exist = chercher(table,partie,sortie);
                if (exist == 0)
                        concat(ascii,sortie);
                else
                      {
                        *erreur = 1;
                        ct = 0;
                      }
                nbre = nbre - 2;
           }
          if (*erreur == 0)
           {
                if (pos1 != longu)
                      {
                        prendre(localpart,pos1+1,longu,partie);
                        concat(ascii,partie);
```

```
                    }
                car = '_';
                car1 = ' ';
                replace(ascii,car,car1);
            }
        }
    }
if (ct == 0)
    {
      creer(ascii);
      copier(localpart,ascii);
    }

}


/* Primitive3 mapdomX400 */
/* --------------------- */


mapdomX400(domain,X400)

char domain[maxadr],X400[maxadr];

{
int pos1,exist,longu,i,c,a,p,o,lg,nbre;
char car,table[maxadr],sortie[maxadr],entree[maxadr],valeur[maxadr];

pos1 = 1;
i= 0;
table[0] = 'r';
table[1] = 'a';
table[2] = 'r';
table[3] = 'e';
table[4] = '.';
table[5] = 'c';
table[6] = '\0';
longu = longueur(domain);
car = '.';
nbre = nbrecar(domain,car,pos1,longu);
exist = chercher(table,domain,sortie);
while ( exist != 0 && i != nbre)
      {
         car = '.';
         pos1 = cherchercar(domain,pos1,car);
         prendre(domain,pos1+1,longu,entree);
         exist = chercher(table,entree,sortie);
         i = i +1;
      }
if ( exist == 1)
      {
         creer(entree);
         exist = chercher(table,entree,sortie);
         pos1 = longu+1;
         i = i + 1;
      }
if (pos1 != 1)
      {
         if ( exist == 0)
             aattribut(sortie,&c,&a,&p,&o);
```

```
            else
              {
                c = 0;
                a = 0;
                p = 0;
                o = 0;
                creer(sortie);
              }
            creer(X400);
            while (i != 0)
              {
                i = i - 1;
                prendre(domain,1,posi-1,entree);
                lg = longueur(entree);
                car = '.';
                posi = dernier car(entree,car,lg);
                prendre(entree,posi+1,lg,valeur);
                if (c == 0)
                   {
                     lg = longueur(X400);
                     car = 'C';
                     ajout(X400,car,lg);
                     c = 1;
                   }
                else if (a == 0)
                   {
                     lg = longueur(X400);
                     car = 'A';
                     ajout(X400,car,lg);
                     a = 1;
                   }
                else if ( p == 0)
                   {
                     lg = longueur(X400);
                     car = 'P';
                     ajout(X400,car,lg);
                     p = 1;
                   }
                else if (o == 0)
                   {
                     lg = longueur(X400);
                     car = 'O';
                     ajout(X400,car,lg);
                     o = 1;
                   }
                else
                   {
                     lg = longueur(X400);
                     car = 'O';
                     ajout(X400,car,lg);
                     car = 'U';
                     lg = lg +1;
                     ajout(X400,car,lg);
                   }
                car = '=';
                lg = longueur(X400);
                ajout(X400,car,lg);
                concat(X400,valeur);
                lg = longueur(X400);
                car = ';';
                ajout(X400,car,lg);
              }
            concat(X400,sortie);
        }
else
        {
            creer(X400);
            copier(sortie,X400);
```

```
/* Primitive4 mapencoded */
/* --------------------- */

mapencoded(encoded,X400)

char encoded[maxadr],X400[maxadr];

{
int pos1,pos2,longu,lg;
char car,partie[maxadr];

longu = longueur(encoded);
pos1 = 0;
creer(X400);
car = '.';
pos2 = chercnercar(encoded,pos1,car);
if (pos2 != 0)
    {
      if ((pos2-pos1)>2)
        {
          car = 'G';
          ajout(X400,car,0);
          car = 'I';
          ajout(X400,car,1);
          car = '=';
          ajout(X400,car,2);
          prendre(encoded,pos1+1,pos2-1,partie);
          concat(X400,partie);
          lg = longueur(X400);
          car = ';';
          ajout(X400,car,lg);
          pos1 = pos2;
          car = '.';
          pos2 = cherchercar(encoded,pos1,car);
        }
      if (pos2 -pos1 == 2)
        {
          lg = longueur(X400);
          car = 'I';
          ajout(X400,car,lg);
          car = '=';
          ajout(X400,car,lg+1);
          prendre(encoded,pos1+1,pos2-1,partie);
          concat(X400,partie);
          lg = longueur(X400);
          car = ';';
          ajout(X400,car,lg);
          pos1 = pos2;
          car = '.';
          pos2 = cherchercar(encoded,pos1,car);
          while (pos2 != 0 && pos2-pos1 == 2)
           {
             prendre(encoded,pos1+1,pos2-1,partie);
             lg = longueur(X400);
             inserer(X400,partie,lg);
             pos1 = pos2;
             car = '.';
             pos2 = chercnercar(encoded,pos1,car);
           }
        }
    }
```

```
lg = longueur(X400);
car = '5';
ajout(X400,car,lg);
car = '=';
ajout(X400,car,lg+1);
prendre(encoded,pos1+1,longu,partie);
concat(X400,partie);
lg = longueur(X400);
car = ';';
ajout(X400,car,lg);
}


/* Primitive5 : maporname */
/* ---------------------- */


maporname(stdorname,X400)

char stdorname[maxadr],X400[maxadr];

{
int pos1,pos2,nbre,longu,egal,stop,pos3;
char car,aux,inter[maxadr],quoted[maxadr],car1,attribute[maxadr];

creer(X400);
longu = longueur(stdorname);
pos1 =0;
aux = '/';
pos2 = 1;
nbre = nbrecar(stdorname,aux,pos2,longu);
while (nbre != 0)
   {
      aux = '/';
      pos2 = cherchercar(stdorname,pos1,aux);
      if (pos2 > 5)
         {
           inter[0] = '(';
           inter[1] = '0';
           inter[2] = '3';
           inter[3] = '6';
           inter[4] = ')';
           inter[5] = '\0';
           prendre(stdorname,pos2-5,pos2-1,quoted);
           egal = comparer(quoted,inter);
           if (egal == 0)
             {
               aux = '$';
               changer(stdorname,pos2,aux);
             }
         }
      nbre = nbre - 1;
      pos1 = pos2;
   }
pos1 =0;
aux = '=';
pos2 = 1;
nbre = nbrecar(stdorname,aux,pos2,longu);
while (nbre != 0)
   {
      aux = '=';
      pos2 = cherchercar(stdorname,pos1,aux);
      if (pos2 > 5)
         {
```

```
            inter[0] = '(';
            inter[1] = '0';
            inter[2] = '3';
            inter[3] = '6';
            inter[4] = ')';
            inter[5] = '\0';
            prendre(stdorname,pos2-5,pos2-1,quoted);
            egal = comparer(quoted,inter);
            if (egal == 0)
              {
                aux = '%';
                changer(stdorname,pos2,aux);
              }
          }
        nbre = nbre - 1;
        pos1 = pos2;
    }
car = '/';
car1 = ';';
replace(stdorname,car,car1);
stop = 0;
pos1 = 1;
while (pos1 != longu )
    {
        car = '=';
        pos2 = cherchercar(stdorname,pos1,car);
        car = ';';
        pos3 = cherchercar(stdorname,pos2,car);
        if (pos2 - pos1 == 3)
            {
              prendre(stdorname,pos1+1,pos2-1,quoted);
              inter[0] = 'P';
              inter[1] = 'N';
              inter[2] = '\0';
              egal = comparer(quoted,inter);
              if (egal == 0)
                {
                  if (pos3 -pos2 >1)
                    {

                      prendre(stdorname,pos2+1,pos3-1,quoted);
                      mapencoded(quoted,attribute);
                      concat(X400,attribute);
                    }
                  stop = 1;
                }
            }
        if (stop != 1)
            {
              prendre(stdorname,pos1+1,pos3,quoted);
              concat(X400,quoted);
            }
        else
              stop = 0;
        pos1 = pos3;
    }
car = ';';
car1 = '/';
replace(stdorname,car,car1);
car = '$';
car1 = '/';
replace(stdorname,car,car1);
replace(X400,car,car1);
car = '%';
car1 = '=';
replace(stdorname,car,car1);
```

```
error(val)

int val;

{

FILE *result;


result = fopen("/users/students/vfa/resultat.dat","r");
if (result == NULL)
 result = fopen("/users/students/vfa/resultat.dat","w");
else
 result = fopen("/users/students/vfa/resultat.dat","a");

switch (val) {

  case 1  : fprintf(result,"absence d'un caractere @ dans l'adresse \n");
            break;

  case 2  : fprintf(result,"un caractere @ est en derniere position dans
             l'adresse \n");
            break;

  case 3  : fprintf(result,"pas de caractere @ pour determiner le domaine \n");
            break;

  case 4  : fprintf(result,"plusieurs caracteres @ determinent une route \n");
            fprintf(result,"mais aucun n'est en premiere position \n");
            break;

  case 5  : fprintf(result,"caractere different d'une lettre, d'un chiffre\n");
            fprintf(result,"d'un trait d'union ou d'un point dans \n");
            fprintf(result,"la partie domaine de l'adresse \n");

  case 6  : fprintf(result,"le premier caractere d'un sous-domaine est
             different d'une lettre \n");
            break;

  case 7  : fprintf(result,"le dernier caractere d'un sous-domaine est \n");
            fprintf(result,"different d'une lettre ou d'un chiffre \n");
            break;

  case 8  : fprintf(result,"absence d'un sous-domaine \n");
            break;

  case 9  : fprintf(result,"pas de correspondance d'un caractere ASCII en
            printablestring \n");
            break;

case 10 : fprintf(result,"si la partie locale est en syntaxe std-ornane, \n");
            fprintf(result,"le premier caractere doit etre / \n");
            break;
```

```
case 11 : fprintf(result,"si la partie locale est en syntaxe std-ornane, \n");
          fprintf(result,"le dernier caractere doit etre / \n");
           break;

case 12 : fprintf(result,"si la partie locale est en syntaxe std-ornane, \n");
          fprintf(result,"un seul caractere = doit se trouver entre deux /\n");
           break;

case 13 : fprintf(result,"si la partie locale est en syntaxe std-ornane, \n");
          fprintf(result,"un attribut n'est pas defini \n");
          break;

case 14 : fprintf(result,"si la partie locale est en syntaxe std-ornane, \n");
          fprintf(result,"la valeur de l'attribut PN est incorrecte \n");
           break;

case 15 : fprintf(result,"si la partie locale est en syntaxe encoded-on, \n");
          fprintf(result,"le terme initial est incorrect \n");
           break;

case 16 : fprintf(result,"si la partie locale est en syntaxe encoded-on, \n");
          fprintf(result,"le terme givenname est incorrect \n");
           break;

case 17 : fprintf(result,"si la partie locale est en syntaxe encoded-on, \n");
          fprintf(result,"le terme surname est incorrect \n");
           break;

case 18 : fprintf(result,"si la partie locale est en syntaxe encoded-on, \n");
          fprintf(result,"absence d'un surname \n");
           break;

case 19 : fprintf(result,"ORName avec deux attributs identiques \n");
          break;

case 20 : fprintf(result,"forme incorrecte d'ORName \n");
          break;

case 21 : fprintf(result,"valeur incorrecte d'attribut dans l'ORName \n");
          break;

case 22 : fprintf(result,"pas d'attribut specifiant la passerelle \n");
          break;

case 23 : fprintf(result,"absence du caractere : pour delimiter la route\n");
          break;

case 24 : fprintf(result,"absence de la partie adresse specifique dans
           l'adresse \n");
          break;

case 25 : fprintf(result,"la syntaxe de l'adresse specifique est erronee\n");
          break;
```

```
case 26 : fprintf(result,"la syntaxe du domaine de l'adresse specifique est
          erronee \n");
          break;

case 27 : fprintf(result,"la syntaxe de la partie locale de l'adresse specifiqu
          est erronee \n");
          break;

case 28 : fprintf(result,"absence d'un domaine dans la partie route \n");
          break;

case 29 : fprintf(result,"la syntaxe d'un domaine dans la partie route est
          erronee \n");
          break;

case 30 : fprintf(result,"absence d'un caractere , a la fin d'un domaine
          de la partie route \n");
          break;

case 31 : fprintf(result,"un 822-P1 Recipient sous forme route-address
          est rejete \n");
          break;

default : break;

          }

fclose(result);
```

BIBLIOGRAPHIE.

[Allman E., Sendmail] :

- ERIC ALLMAN, "Sendmail - An Internetwork Mail Router", in UNIX System Manager's Manual, 4.2 Berkeley Software Distribution, Virtual VAX-11 Version, Computer Science Division, Department of Electrical Engineering and Computer Sciencen University of California, Berkeley, CA (Mar. 1984), as reprinted by the USENIX Association.

- ERIC ALLMAN, "Sendmail - Installation and Operation Guide Version 4.2", in UNIX System Manager'sManual, 4.2 Berkeley Software Distribution, Virtual VAX-11 Version, Computer Science Division, Departement of Electrical Engineering and Computer Science, University of California, Berkeley, CA (Mar. 1984), as reprinted by the USENIX Association.

[CCITT, X400] :

Recommandations X400, Data Communication Networks Message Handling Systems, Red Book VIIIth plenary assembly Malaga Torremolinos 8-19 October 1984, Genève 1985.

[CCITT, X409] :

Recommandations X409, Data Communication Networks Message Handling Systems : Presentation Transfer Syntax and Notation, Red Book VIIIth plenary assembly Malaga Torremolinos 8-19 October 1984, Genève 1985.

[CCITT, X411] :

Recommandations X411, Data Communication Networks Message Handling Systems : Message Transfer Layer, Red Book VIIIth plenary assembly Malaga Torremolinos 8-19 October 1984, Genève 1985.

[CCITT, X420] :

Recommandations X420, Data Communication Networks Message Handling Systems : Interpersonal Messaging User Agent Layer, Red Book VIIIth plenary assembly Malaga Torremolinos 8-19 October 1984, Genève 1985.

[Crocker D.H., RFC-822] :

DAVID.H. CROCKER, "Standard of the Format of ARPA Internet Messages", RFC-822, August 1982.

[Grimm R.] :

RUEDIGER GRIMM, "A Minimum Profile for RFC-987 : Mapping between addresses in RFC-822 format and X400 standard attributes", december 1987.

[Kille S., RFC-987] :

STEVE KILLE, "Mapping Between X400 and RFC-822", RFC-987, june 1987.

[Postel J., RFC-821] :

J. POSTEL, "Simple Mail Transfer Protocol", RFC-821, october 1982.

[Postel, Reynolds, RFC-920] :

J.POSTEL & J. REYNOLDS, "Domain Requirements", RFC-920, october 1984.

[Tanenbaum A.] :

A. TANENBAUM, "Networks Protocols", Computing Surveys, Vol 13, N° 4, december 1981.

[van Lamsweerde A.] :

ALEX VAN LAMSWEERDE, Notes du cours de Méthodologie de Développement de Logiciels, 2ème licence en informatique, Facultés Notre-Dame de la Paix Namur, 1987-88.