Network Working Group
Request for Comments: 2060
Rends obsolète: 1730
Catégorie: Standards Track

M. Crispin
Université de Washington
Décembre 1996

PROTOCOLE D'ACCES AUX MESSAGES INTERNET - VERSION 4 rev1

(INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1)

- **Statut de ce mémo**

  On spécifie par ce document un protocole Internet de niveau standard track pour la communauté Internet, ainsi que les demandes de discussions et de suggestions d'améliorations. Si vous le souhaitez, vous pouvez vous référer à l'actuelle édition de l 'Internet Official Protocol Standards" (STD 1) pour connaître l'état de standardisation et le statut pour ce protocole. La distribution de ce mémo n'est pas limitée.

- **Abrégé**

  L'Internet Message Access Protocol, Version 4rev1 (IMAP4rev1) permet à un client d'accéder et de manipuler des messages de courrier électronique sur un serveur. IMAP4rev1 autorise la manipulation de dossiers distants contenant des messages, appelés "boîte aux lettres", qui d'une certaine façon sont fonctionnellement équivalents à des boîtes aux lettres locales. IMAP4rev1 donne aussi la possibilité pour un client hors ligne de se resynchroniser avec le serveur (voir aussi [IMAP-DISC]).

  IMAP4rev1 comprends des opérations pour la création, la suppression, le renommage de boîte aux lettres, pour permettre de vérifier l'arrivée de nouveaux messages, pour l'enlèvement permanent de messages, pour la mise en place et la suppression de drapeaux, pour l'analyse syntaxique vue dans [RFC-822] et [MIME-IMB], pour la recherche et une sélection attrayante d'attribue de message, de texte, et portions de ceux-ci. Les messages en IMAP4rev1 sont accessibles par l'utilisation de nombres. Ces nombres sont soit, des numéros de séquences de messages, soit des identifiants uniques.

  IMAP4rev1 supporte un seul et simple serveur. Un mécanisme d'information d'accès à la configuration pour supporter de multiples serveurs IMAP4rev1 est en cours de discussion dans [ACAP].

  IMAP4rev1 ne définit pas la façon de poster le courrier ; Cette fonction est traitée par un protocole

de transfert de courrier tel que [SMTP].

IMAP4rev1 est conçu pour être compatible de manière ascendante avec les protocoles [IMAP2] et le non publié IMAP2bis. Au cours de l'évolution de IMAP4rev1, certains aspects du protocole originel sont devenus obsolètes. Les commandes, les réponses, et les formes de données obsolètes, qu'une implémentation IMAP4rev1 peut rencontrer quand on utilise une implémentation des premiers temps sont décrites dans [IMAP-OBSOLETE].

Les autres problèmes de compatibilité avec IMAP2bis, la variante la plus commune par rapport au protocole premier, sont débattus dans [IMAP-COMPAT]. Le débat complet sur les problèmes de compatibilité avec les rares variantes de [IMAP2] se trouve dans [IMAP-HISTORICAL] ; Ce document relève, à l'origine, d'un intérêt historique.

- **Table des Matières**

- **Spécification du protocole IMAP4rev1**

- **1. Comment lire ce document**

- 1.1. Organisation de ce document

  Ce document est écrit du point de vue de l'implémentation d'un serveur ou d'un client IMAP4rev1. Au-delà du survol du protocole en section 2, ce n'est pas optimisé pour quelqu'un qui essaierait de comprendre les opérations du protocole. La matière située de la section 3 à la section 5 donne le contexte général et les définitions avec lesquels IMap4rev agit.

  Les sections 6, 7, et 9 décrivent respectivement les commandes, les réponses, et la syntaxe IMAP. Les relations entre elles sont telles qu'il est presque impossible de les comprendre séparément. Plus pr&eacutecisement, ne vous attendez pas à déduire la syntaxe des commandes d'une seule section ; Reportez-vous plut&ocirct à la section de la syntaxe formelle.

- 1.2. Conventions utilisées dans ce document

  Dans les exemples, "C:" et "S:" indiquent respectivement les lignes envoyées par le client et le serveur.

Les termes qui suivent sont utilisés dans ce document pour annoncer les conditions de cette spécification.

1) DOIT (MUST), ou l'adjectif NECCESSAIRE (REQUIRED), signifie que la définition est une nécessité absolue à la spécification.

2) NE DOIT PAS (MUST NOT) signifie que la définition est absolument prohibée dans la spécification.

3) DEVRAIT (SHOULD) signifie qu'il peut exister des raisons valables dans des circonstances spéciales pour ignorer un cas particulier, mais la totalité des implications doivent (MUST) être comprise et pesée avec attention avant de choisir une direction différente.

4) NE DEVRAIT PAS (SHOULD NOT) signifie qu'il pourrait exister des raisons valables dans des circonstances particulières quand un comportement particulier est acceptable ou même utile. Mais la totalité des implications DEVRA (SHOULD) être comprise et le cas devra être méticuleusement soupeser avant de mettre en pratique des comportements décrient avec cette marque

5) PEUT (MAY), ou l'adjectif OPTIONNEL (OPTIONAL), signifie qu'un article est vraiment optionnel. Un vendeur peut choisir d'inclure l'option parce qu'une certaine partie de marché en a besoin ou parce que le vendeur sent que cela améliorera le produit alors qu'un autre vendeur peut omettre la même option. Une implémentation qui n'inclut pas une option particulière DOIT être préparée a être interopératif avec un autre système qui a cette option.

   "POSSIBLE" ("Can") est utilisé à la place de "PEUT" ("may"), quand on fait référence à des circonstances ou à des situations éventuelles, à l'opposée a une possibilité optionnelle du protocole.

   "Utilisateur" ("User") est utilisé pour faire référence à un utilisateur humain, tandis que "Client" fait référence au logiciel que fait fonctionner l'utilisateur.

   "Connexion" fait référence à l'interaction de la séquence complète client/serveur, de l'établissement initial de la connexion réseau jusqu'à ce quelle se termine. "Sessions" fait référence à la séquence d'interaction client/serveur à partir de l'instant où une boite à lettre est sélectionnée (commande SELECT ou EXAMINE) jusqu'à ce que l'instant où la sélection se termine (SELECT ou EXAMINE d'une autre boite à lettre, commande CLOSE ou terminaison de connexion).

   Les caractères sont les 7-bit US-ASCII à moins que d'autres soient spécifiés. D'autres jeux de caractère sont indiqués en utilisant "CHARSET", comme il est décrit dans [MIME-IMT] et défini dans [CHARSET]. Les CHARSET ont une sémantique importante en plus du jeu de caractères défini; Referez vous à ces documents pour plus de détails.

- **2. Vue d'ensemble du protocole**

- 2.1. Niveau lien

Le protocole IMAP4rev1 présuppose d'avoir un flux de donnée fiable tel que celui fournit par TCP.

Quand TCP est utilisé, un serveur IMAPrev1 écoute sur le port 143.

- 2.2. Commandes et Réponses

  Une connexion IMAP4rev1 consiste en l'établissement d'une connexion réseau client/serveur, à la bienvenue de la part du serveur, et à des interactions client/serveur. Ces interactions client/serveurs consistent en commandes clients, en données venant du serveur, et en réponses de résultat d'achèvement.

  Toutes les interactions transmissent par le client et le serveur sont sous la forme de lignes ; C'est à dire, en des chaînes de caractères qui finissent par un CRLF. Le receveur de protocole d'un client ou d'un serveur IMAP4rev1 lit soit une ligne, soit en une séquence d'octets avec un comptage connu suivi par une ligne.

- 2.2.1. Expéditeur de protocole client et receveur de protocole serveur

  Les commandes clients commencent par une opération. Chaque commande client est préfixée par un identifiant (Typiquement, une chaîne alphanumérique courte par ex. A0001, A0002, etc.;) appelé un "tag" (marqueur ou drapeau). Un drapeau différent est généré par le client à chaque commande.

  Il y a deux cas pour lesquels une ligne générée par le client ne représente pas une commande entière. Pour un cas, un argument de commande est mis entre cotes avec un compteur d'octet (voir la description d'un littéral dans une chaîne sous le format d'une donnée) ; Pour l'autre cas, les arguments de la commande nécessite une réponse en retour (voir la commande AUTHENTICATE). Dans tous les cas, le serveur envoie une réponse de demande de continuation de commande si c'est prêt pour les octets (si c'est approprié) et pour le reste de la commande. Cette réponse est préfixée par le marqueur "+".

  Remarque: Si, au lieu de cela, le serveur détecte une erreur dans la commande, il envoie une réponse de mauvaise terminaison (BAD) avec un drapeau entourant (matching) la commande (comme décrit au-dessous) pour rejeter la commande et empêcher le client qu'il envoie le restant de la commande.

  Il est aussi possible pour le serveur d'envoyer une réponse de terminaison pour d'autres commandes (si de multiple commande sont en cours), ou d'envoyer des données non mises entre marqueur. Dans tout les cas, la requête de continuation de commande est toujours en attente ; le client DOIT envoyer une commande complète (tout en recevant les réponses de demande de commande de continuation et de commande de continuation pour la commande) avant d'initier une nouvelle commande.

  Le receveur de protocole d'un serveur IMAP4rev1 lit une ligne de commande du client, analyse la commande et ses arguments, et transmet les données serveurs et la terminaison de commande de serveur terminant la réponse.

- 2.2.2. Expéditeur de protocole serveur et receveur de protocole client

  Les données transmissent par le serveur vers le client et les réponses de statut qui n'indique pas la fin d'une commande sont préfixées par le drapeau "*" et on appelle cela, des réponses non marquées("untagged").

  Les données d'un serveur PEUVENT (MAY) être envoyé en tant que résultat d'une commande client, ou PEUVENT (MAY) être envoyé unilatéralement par le serveur. Il n'y a pas de différence syntaxique entre des données de serveur qui résulterait d'une commande spécifique et des données serveur qui serait envoyées unilatéralement.

Les réponses serveur de résultat de fin indique le succès ou l'échec de l'opération. Elle est marquée par le même indicateur qu'il y avait sur la commande client qui avait commencé l'opération. De cette façon, si plus d'une commande est en cours, l'indicateur de la réponse serveur de fin identifie la commande pour laquelle la réponse s'applique. Il y a trois réponses serveur de fin possible : OK (indiquant le succès), NO (indiquant l'échec), ou BAD (indiquant une erreur de protocole comme une commande non reconnue ou une erreur de syntaxe).

Le receveur de protocole d'un client IMAP4v1 lit une réponse constituée par une ligne venant du serveur. Il fait alors des actions sur les réponses en se basant sur le premier drapeau de la réponse, qui peut être un marqueur "*" ou un "+". Un client doit être préparer pour accepter à tout moment, n'importe quelle réponse de serveur. Cela peut comprendre des données de serveur qui ne sont pas demandée. Les données de serveur DOIVENT (should) être enregistrée, de manière à ce que le client puisse faire référence à la copie enregistrée plutôt que d'envoyer une commande au serveur pour demander les données. Dans le cas de certaine données serveur, les données DOIVENT (MUST) être enregistrée. On parle de ce sujet plus en détail dans la section réponses de serveur.

- 2.3. Attributs de message

  En plus du texte des messages, chaque message a plusieurs attributs associés. Ces attributs peuvent être récupérés individuellement ou en conjonction avec d'autres attributs ou avec des textes de messages.

- 2.3.1. Numéros de messages

  Les messages en IMAP4rev1 sont accessibles par un ou deux nombres. L'identifiant unique et le numéro de séquence du message.

- 2.3.1.1. Attribut de message Identification Unique (UID)

  Une valeur sur 32-bit est assigné à chaque message, qui quand elle est utilisée avec la valeur de validité d'identification unique (voir ci-dessous) forme une valeur sur 64-bits, qui garantie en permanence de ne pas se référer à n'importe quel autre message situé dans la boîte aux lettres. Les identifiants uniques sont affectés de manière strictement croissante dans la boîte à lettre. De ce fait, à chaque message ajouté à la boîte aux lettres, on affecte un UID plus grand que celui qui avait été ajouté précédemment.

  Contrairement aux numéros de séquence de message, les identifiants uniques ne sont pas nécessairement contiguë. Les identifiants uniques sont persistant aussi au travers des différentes sessions. Cela permet à un client de resynchroniser son état à partir de la session précédante avec le serveur (voir accès client déconnecté ou offline) ; Ceci est traité plus tard dans [IMAP-DISC].

  On associe à chaque boîte aux lettres une valeur de validité d'identification unique, qui est envoyée dans un code de réponse UIDVALIDITY situé dans la réponse OK de type non marqué au moment de la sélection de la boîte aux lettres. Si les identifiants uniques d'une session précédente échouent lors de la tentative de continuation de la session actuelle, la valeur de validité de l'identifiant unique DOIT (MUST) être plus grande que celle qui a été utilisé dans la session la plus récente.

  Remarque: Les identifiants uniques DOIVENT (MUST) être à tout moments strictement croissant. Si le stockage physique du message est réordonné par un agent non-IMAP, cela nécessite que les identifiants uniques dans la boite à lettre soit régénérer puisque les identifiants uniques précédents ne sont plus strictement ascendant du fait du résultat de la remise en ordre. Un autre exemple pour lequel les identifiants uniques doivent être régénérer, est si le stockage des messages n'a pas de mécanisme pour stocker des identifiants uniques. Bien que cette spécification l'admette, se peut être inévitable dans des environnements de serveurs sans panne, il est FORTEMENT

ENCOURAGE d'appliquer des techniques de stockage de message qui évite ce problème (de régénération).

Une autre cause de non-persistance est que si la boîte aux lettres est supprimée et d'une autre boîte aux lettres portant le même nom est crée à une date ultérieure, puisque le nom est le même, un client peut ne pas savoir que c'est une nouvelle boite à lettre à moins que la valeur de validité d'identifiant unique soit différente. Une bonne valeur a utilisé pour la valeur de validité l'identifiant unique est une représentation sur 32 bits de la date/heure de création de la boite à lettre. C'est juste d'utiliser une constante tel que 1, mais seulement si elle garantit que les identifiants uniques ne seront jamais plus réutilisé, même dans le cas ou une boite à lettre est supprimé (ou est renommé) et qu'une nouvelle boîte aux lettres avec le même nom soit crée quelque temps plus tard.

L'identifiant unique d'un message NE DOIT PAS (MUST NOT) changer pendant la session, et NE DEVRAIT PAS (SHOULD NOT) changer entre les sessions. S'il n'est pas possible de préserver l'identifiant unique lors d'une session suivante, chaque session suivante DEVRA ABSOLUMENT (MUST) avoir une nouvelle valeur de validité unique d'identifiant unique qui sera plus grande que n'importe quelle valeur utilisée précédemment.

- 2.3.1.2. Attribut du message le numéro de séquence de message

  C'est la position relative dans la boîte aux lettres qui va de 1 à la valeur du nombre de message. Cette position DOIT (MUST) être ordonné par ordre croissant d'identifiants unique. Pour chaque nouveau message qui est ajouté, on affecte un numéro de séquence de message qui est plus grand de 1 que le nombre de message dans la boîte aux lettres avant que ce nouveau message soit ajouté ;

  Les numéros de séquence de message peuvent être réattribué pendant la session. Par exemple, quand un message est enlevé de façon durable (supprimé) de la boîte aux lettres, le numéro de séquence de tous les numéros de séquence de message de tous les messages suivants sont décrémenté. De la même façon, on peut affecter à un nouveau message un numéro de séquence de message qui était jadis tenu par un quelconque autre message précédant qui a été supprimé.

  En plus de l'accès aux messages par la position relative dans la boîte aux lettres, les numéros des séquences des messages peuvent être utilisés dans des calculs mathématiques. Par exemple, si un "EXISTS 11" sans "tag" est reçu et que précédemment un "8 EXISTS" sans "tag" soit déjà arrivé, trois nouveaux messages sont arrivés en portant les numéros de séquence 9,10 et 11. Un autre exemple. Si le message 287 dans une boîte aux lettres de 523 messages avait l'UID 12345, il y exactement 286 messages qui ont des IUD plus petits et 236 messages qui ont de plus grand UID.

- 2.3.2. Drapeaux attribut de message

  Liste composée de zéro ou de plusieurs de marqueur ayant un nom et qui est associé au message. Un drapeau est activé par son ajout à cette liste, et il est désactivé par sa suppression. Il y a deux types de marqueur sous IMAP4rev1. Un drapeau peut, soit être de type permanent, soit de type session.

  Un drapeau système est un drapeau ayant un nom qui est prédéfini dans cette spécification. Tous les drapeaux systèmes commencent par "\". Les drapeaux système d'état certain (\deleted et \seen) ont une sémantique spéciale qui est décrite ailleurs. Les drapeaux systèmes définis actuellement sont :

  \Seen Le message a été lu

\Answered On a répondu au message

\Flagged Le message est "flagged" pour y donner une attention urgente/spéciale

\Deleted Le message est "deleted" (supprimé) pour que plus tard un EXPUNGE puisse l'enlever

\Draft Le message n'a pas été entièrement composé (marqué en tant que brouillon (draft)).

\Recent Le message est arrivé récemment dans cette boîte aux lettres. Cette session est la première session qui ait reçu une notification a propos de ce message.

Les sessions ultérieures ne verront pas l'état \Recent pour ce message. Ce drapeau ne peut être modifié par le client.

S'il n'est pas possible de déterminer si oui ou non, cette session est la première session a être notifiée du message, alors ce message DEVRA (SHOULD) être considéré comme récent. Si de multiples connexions ont sélectionné la même boîte aux lettres simultanément, on ne peut définir laquelle de ce ses connexions verra les messages arrivés nouvellement avec l'état \Recent et quelles vont être celles qui le verront sans \Recent.

Un mot clef peut être défini par l'implémentation serveur. Les mots clefs ne commencent pas par "\". Les serveurs PEUVENT (MAY) permettre aux clients de définir de nouveaux mots clefs dans la boîte aux lettres (pour plus d'information, voir la description du code de réponse PERMANENTFLAGS).

Un drapeau peut être permanent ou session dans le système basé sur les drapeaux. Les drapeaux permanents sont ceux pour lesquels le client peut en permanence rajouter ou enlever les drapeaux du message ; C'est à dire, que les sessions suivantes ne verront aucun changement pour les drapeaux permanents. Les changements pour les drapeaux de sessions ne sont valides que pour la session.

Remarque: Le drapeau système \Recent est un cas spécial de drapeaux de session. \Recent ne peut être utilisé en tant qu'argument de la commande STORE, et ainsi peut ne pas être changé du tout (at all).

- 2.3.3. Attribut de date interne d'un message

  C'est la date et l'heure interne d'un message sur le serveur. Ce n'est pas la date et l'heure de l en-tête de la [RFC-822], mais plutôt une date et heure qui reflète quand le message a été reçu. Dans le cas de messages qui ont été délivré par [SMTP], DEVRAIT (SHOULD) être la date et l'heure de livraison finale du message comme c'est défini par [SMTP]. Dans le cas de messages délivrés par une commande IMAP4rev1 COPY, se DEVRAIT (SHOULD) être la date et l'heure interne du message source. Dans le cas de messages délivrés par une commande IMAP4rev1 APPEND, se DEVRAIT (SHOULD) être la date et l'heure qui est spécifié dans la description de la commande APPEND. Dans tous les autres cas, c'est défini par l'implémentation.

- 2.3.4. Attribut de taille du message de la [RFC-822]

  C'est le nombre d'octet dans un message, comme cela s'exprime dans le format de la [RFC-822].

- 2.3.5. Attribut de structure d'enveloppe d'un message

C'est une représentation syntaxique [parsed] de l'information d'enveloppe [RFC-822] (ne pas faire la confusion avec l'enveloppe [SMTP]) du message.

- 2.3.6. Attribut de structure d'un corps d'un message

  C'est une représentation syntaxique [parsed] de l'information de la structure du corps [MIME-IMB] du message.

- 2.4. Texte d'un message

  En plus d'être capable d'aller chercher le texte [RFC-822] entier d'un message, IMAPrev1 permet l'aller chercher des portions de texte sur la totalité du message. En particulier, il est possible de récupérer les en-têtes [RFC-822] de message, le corps de message [RFC-822], la partie du corps [RFC-822], ou un en-tête [MIME-IMB].

- **3. Diagramme de flux et d'état**

  Un serveur IMAP4rev1 est toujours dans un des quatre états possibles. La plupart des commandes ne sont valable que pour seulement un des états bien définis. Il y a une erreur de protocole pour le client qui attends une commande alors que la commande n'est pas dans un état approprié. Dans ce cas, le serveur répondra avec un résultat BAD ou NO de terminaison de commande (ceci dépendant de l'implémentation).

- 3.1. Etat non authentifié

  Dans un état de non-authentification, le client DOIT (MUST) fournir une justification d'authentification avant que la plupart des commandes puissent être permise. On rentre dans cet état quand une connexion s'établit à moins que la connexion ait été pré-authentifiée.

- 3.2. Etat authentifié

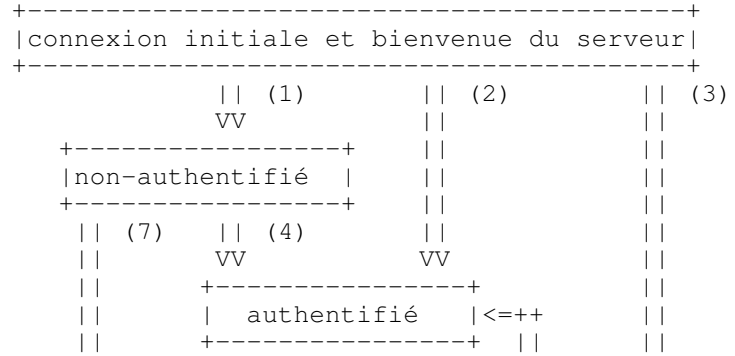
  En état authentifié, le client est authentifié et DOIT (MUST) sélectionner une boîte aux lettres avant que les commandes qui ont des effets sur les messages soient autorisées. On entre dans cet état avec le commencement d'une connexion pré-authentifié, quand des justifications d'authentifications admissibles ont été fournie, ou après une erreur dans la sélection de la boîte aux lettres.

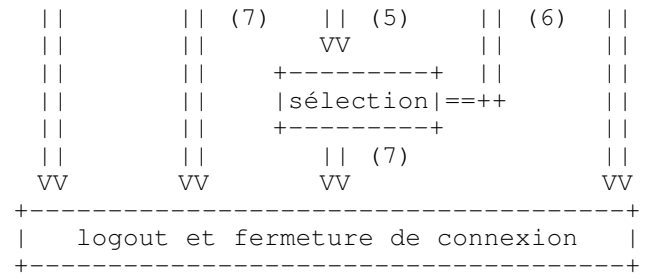
- 3.3. Etat sélectionné

  En état sélectionné, une boîte aux lettres a été sélectionnée pour y accéder. On entre dans cet état quand on a sélectionné avec succès la boîte aux lettres.

- 3.4. Etat de logout (sortie)

  En état de logout, la connexion a été terminée, et le serveur va fermer la connexion. On peut entrer dans cet état, par le résultat d'une requête client ou par une décision unilatérale du serveur.





(1) connexion sans pré-authentification (bienvenue OK)
(2) connexion pré-authentifié (Bienvenue PREAUTH)
(3) connexion rejetée (bienvenue BYE)
(4) commande de LOGIN ou AUTHENTICATE réussie
(5) commande SELECT ou EXAMINE réussie
(6) commande CLOSE, ou commande SELECT ou EXAMINE échoué
(7) commande LOGOUT, arrêt du serveur, ou connexion fermée

- **4. Format des données**

  IMAP4rev1 utilise des commandes et des réponses sous forme de texte. Les données en IMAP4rev1 peuvent être sous une des multiples formes : atome, nombre, chaîne, liste mise entre parenthèse ou NIL.

- 4.1. Atome

  Un atome est composé par un ou plusieurs caractères non-spéciaux.

- 4.2. Nombre

  Un nombre consiste en un ou plusieurs caractère, et il représente une valeur numérique.

- 4.3. Chaîne

  Une chaîne est sous une des deux formes : un littéral ou une chaîne mise entre double cote ("quote"). La chaîne mise entre "quote" est une alternative qui évite la surcharge du fait du traitement d'un littéral aux prix de la limitation des caractères qui peuvent être utilisés dans une chaîne mise entre cote.

  Un littéral est une séquence de zéro ou de plusieurs octets (y compris CR et LF), préfixé par des "quotes" avec un compteur d'octet sous la forme d'une accolade d'ouvrante ('{'), le nombre d'octet, accolade fermante ["}"], et CRLF. Dans le cas de littéraux transmis du serveur vers le client, le CRLF est immédiatement suivi par les octets de donnée. Dans le cas de littéraux transmis du client vers le serveur, le client DOIT (MUST) attendre pour recevoir une commande de requête de continuation (décrit plus tard dans ce document) avant d'envoyer les données d'octet (et le reste de la commande).

  Une chaîne mise entre "quote" est une séquence de zéro ou plusieurs caractères sur 7-bit en excluant CR et LF, avec les caractères de double "quote" (<">) à chaques bouts.

  La chaîne vide est représentée soit par "" (une chaîne "quotée" avec zéro caractères entre les doubles "quotes") ou par {0} suivi par le CRLF (un littéral avec un nombre d'octet à 0).

  Remarque: Même si le nombre d'octet est zéro, un client en train de transmettre un littéral DOIT (MUST) attendre afin de recevoir une commande de requête de continuation.

- 4.3.1. Chaînes binaires et 8 bits

Le courrier binaire et texte en 8-bit est supporté au travers de l'utilisation d'un contenu [MIME-IMB] utilisant le transfert d'encodage de contenu. Les implémentations IMAP4rev1 PEUVENT (MAY) transmettre des caractères 8-bit ou multit-octet sous forme littérale, mais DEVRAIT (SHOULD) faire cela seulement quand le [CHARSET] est identifié.

Bien qu'un encodage de corps BINARY soit défini, les chaînes binaires non codées (unencoded) ne sont pas permises. Une "chaîne binaire" (binary string), c'est n'importe quelle chaîne avec des caractères NUL. Les implémentations DOIVENT (MUST) encoder les donnée binaires sous une forme textuelle telle que BASE64 avant de transmettre les données. Une chaîne avec une accumulation excessive de caractères CTL POURRAIT (MAY) aussi être considérée comme étant binaire.

- 4.4. Liste entre parenthèse

  Les structures de données sont représentées comme une "liste entre parenthèses ; une séquence d'articles de donnée, délimité par des espaces, et bornées à chacun des bouts par des parenthèses. Une liste entre parenthèse peut contenir d'autres listes entre parenthèses, en utilisant plusieurs niveaux de parenthèses pour indiquer leurs imbrications.

  Une liste vide est représentée par () -- une liste parenthésée avec aucun membre.

- 4.5. NIL

  L'atome spécial NIL représente la non-existence d'un élément de donnée particulier qui est représenté en tant qu'une chaîne ou une liste entre parenthèses, comme étant différente de la chaîne vide "" ou d'une liste entre parenthèse vide ().

- **5. Considérations fonctionnelles**

- 5.1. Nommage des boîtes aux lettres

  L'interprétation des noms de boîte aux lettres est dépendante de l'implémentation. Cependant, le nom de boîte aux lettres INBOX qui est indépendant de la case, est un nom réservé pour signifier "La boîte aux lettres primaire pour cet utilisateur sur ce serveur".

- 5.1.1. Nommage hiérarchique de boîte aux lettres

  Si l'on désire exporter des noms hiérarchiques de boîte aux lettres, les noms de boîte aux lettres DOIVENT (MUST) être hiérarchisé de la gauche vers la droite en utilisant un seul caractère pour séparer les niveaux de hiérarchie. Le même caractère doit être utilisé pour séparer tous les niveaux de hiérarchie pour un même nom.

- 5.1.2. Convention de nommage de l'étendu des noms de boîtes aux lettres

  Par convention, le premier élément hiérarchique de tous noms de boite à lettre qui commence par un caractère '#' identifie l'étendue d'espace du reste du nom. Cela permet de lever l'ambiguïté qui existerait entre différents types de stockage de boîte aux lettres, chacune ayant leur propre nom d'espace. Par exemple, les implémentations qui offrent un accès au newsgroupe USENET POURRAIT (MAY) utiliser le nom d'espace "#news" pour partitionner le nom de l'espace du groupe de travail USENET de celui des autres boites à lettre. Ainsi, le groupe de travail comp.mail.misc aurait le nom de boite à lettre de "#news.comp.mail.misc", et le nommage "comp.mail.misc" pourrait faire référence à un objet diffèrent (par ex. une boîte aux lettres privée d'un utilisateur). (Sic, la traduction du paragraphe 5.1.2. est plus qu'approximative).

- 5.1.3. Convention internationale de nommage des boîtes à lettres

Par convention, les noms internationaux de boîte aux lettres sont spécifiés en utilisant une version modifiée d'encodage UTF-7 décrite dans [UTF-7]. Le but de ces modifications est de corriger les problèmes suivant qu'il y a avec UTF-7 :

1) UTF-7 utilise le caractère "+" pour se déplacer (shifting) ; Cela rentre en conflit avec l'utilisation commune du "+" dans le nom des boîtes aux lettres plus particulièrement les noms de groupe USENET.

2) L'encodage de UTF-7 est BASE64 qui utilise le caractère "/". Cela entre en conflit avec l'utilisation de "/" en tant que délimiteur hiérarchique populaire.

3) UTF-7 interdit l'usage non-encodé de "\" ; Cela entre en conflit avec l'usage de "\" en tant que délimiteur hiérarchique populaire

4) UTF-7 interdit l'usage non-encodé de "~" ; Cela entre conflit avec l'utilisation de "~" pour certain serveur comme indicateur de répertoire "home".

5) UTF-7 permets de multiples formes alternatives pour représenter la même chaîne de caractère. En particulier, les caractères US-ASCII imprimables peuvent être représentés sous une forme encodée.

En modifiant UTF-7, les caractères imprimables US-ASCII, exception faite pour "&", sont représentés par eux-mêmes ; c'est à dire les caractères avec pour valeur d'octet 0x20-0x25 et 0x27-01x7e. Le caractère "&" (0x26) est représenté par la séquence des deux octets "&-".

Tous les autres caractères (octets avec pour valeur 0x00-0x1f, 0x7f-0xff, et tous les octets unicode 16-bits) sont représentés en BASE64 modifié, avec une modification supplémentaire à partir de [UTF-7] qui est que "," est utilisé à la place de "/". Ce qui est modifié BASE64 NE DOIT PAS (MUST NOT) être utilisé pour représenter les caractères US-ASCII qui peuvent s'autoreprésenter.

"&" est utilisé pour shifter (se déplacer) en BASE64 modifié et "-" pour revenir vers les caractères US-ASCII. Tous les noms commencent en US-ASCII, et doivent finir en US-ASCII (cela veut dire, qu'un nom qui finit avec un octet Unicode 16 bit DOIT (MUST) finir par un "-").

Par exemple, voici un nom de boîtes aux lettres qui mixe le texte anglais, japonais, et chinois : ~peter/mail/&ZeVnLIqe-/&U,BTFw-

- 5.2. Dimension des boîtes aux lettres et mise à jour du statut des messages

A n'importe quel moment, un serveur peut envoyer des données que le client n'a pas demandées. Parfois, un tel comportement est NECESSAIRE (REQUIRED). Par exemple, des agents autres que ceux du serveur PEUVENT (MAY) ajouter des messages à la boîte aux lettres (par exemple une distribution de nouveaux courriers), changer les drapeaux de messages dans la boîte aux lettres (par exemple par accès simultané à la boîte aux lettres par plusieurs agents), ou même enlever des messages de la boîte aux lettres. Un serveur DOIT (MUST) envoyer automatiquement la mise à jour concernant la taille de la boîte aux lettres si on observe un changement taille de cette boîte pendant le traitement d'une commande. Un serveur DEVRAIT (SHOULD) envoyer automatiquement des mises à jour de drapeaux sans que cela ne nécessite pour le client de faire une demande explicite pour ce type de mises à jour. Des règles spéciales existent pour la notification par un serveur vers un client en ce qui concerne la suppression de messages pour éviter les erreurs de synchronisations. Pour plus de détail, voir la description de la réponse à EXPUNGE.

Bien que l'on ne doive pas se soucier des décisions d'implémentations qui ont été prises pour qu'un

client puisse se rappeler des données du serveur, une implémentation cliente DOIT (MUST) enregistrer les mises à jours de la taille de la boîte aux lettres. Après la sélection initiale d'une boite à lettre, on NE DOIT PAS (MUST NOT) considérer qu'une commande retournera la taille de la boite à lettre.

- 5.3. Réponse quand aucune commande est en cours

  On permet aux implémentations de serveur d'envoyer une réponse non marquée (exception faite pour EXPUNGE) alors qu'il n'y a pas de commande en cours. Les implémentations de serveurs qui envoient de telles réponses DOIVENT (MUST) tenir compte de la régulation du flux. Plus précisément, ils DOIVENT(MUST) soit (1) vérifier que la taille des données n'excède pas la taille de fenêtre disponible du transport sous-jacent, (2) soit faire des écritures non bloquantes.

- 5.4. Timer Autologout

  Si un serveur a un compteur de temps de sortie automatique, ce compteur de temps DOIT (MUST) doit s'arrêter au bout d'au moins 30 minutes de durée. La réception de N'IMPORTE quelle commande du client durant cet intervalle DEVRAIT (SHOULD) suffire pour relancer le compteur de temps de sortie automatique.

- 5.5. Commandes multiples en cours

  Le client PEUT (MAY) envoyer une autre commande sans attendre la réponse de résultat d'achèvement d'une commande, sujette à des règles d'ambiguïté (voir dessous) et ayant des contraintes de contrôles de flux en ce qui concerne le courant supportant les données. Pareillement, un serveur PEUT (MAY) commencer le traitement d'une autre commande avant que le traitement de la commande en cours soit fini, et qui est exposé à des règles d'ambiguïtés. Cependant, toutes réponses à une requête de continuation et toutes continuations de commande DOIVENT (MUST) être traité avant que la commande suivante soit initiée.

  L'exception apparaît si une ambiguïté pouvait résulter d'une commande qui affecterait le résultat des autres commandes. Les Clients NE DOIVENT PAS (MUST NOT) envoyer de multiples commandes sans attendre, si un résultat ambigu peut subvenir. Si le serveur détecté une possible ambiguïté, il DOIT (MUST) exécuter les commandes pour les terminer dans l'ordre donné par le client.

  Le plus manifeste des exemples d'ambiguïté est quand une commande affecte les résultats d'une autre commande. Par exemple, un FETCH de drapeaux de message et un STORE des ces mêmes drapeaux de messages.

  Une ambiguïté pas évidente peut exister avec des commandes qui permettent une réponse EXPUNGE de type non-marqué (commandes autre que FETCH, STORE, et SEARCH), puisqu'une réponse EXPUNGE non marquée peut rendre non valide les numéros de séquence pour la commande qui suivrait. Ce n'est pas un problème pour les commandes FETCH, STORE, ou SEARCH puisqu'il est interdit aux serveurs d'envoyer des réponses d'EXPUNGE, tant que n'importe de ces commandes sont en cours. Donc, si le client envoie une commande autre que FETCH, STORE ou SEARCH, il doit (MUST) attendre avant d'envoyer une commande avec des numéros de séquence de message.

  Par exemple, les séquences suivantes de commande qui n'attendent pas, sont incorrectes :

  FETCH + NOOP + STORE
  STORE + COPY + FETCH
  COPY + COPY

CHECK + FETCH

Ce qui suit sont des exemples valables de séquences de commande qui n'attendent pas :

FETCH + STORE + SEARCH + CHECK
STORE + COPY + EXPUNGE

- **6. Commandes client**

Les commandes IMAP4rev1 sont décrites dans cette section. Les commandes sont arrangées suivant l'état ou l'on permet de les exécuter. Les commandes qui peuvent être exécuter pour de multiples états sont listées pour l'état minimum où elles sont permises (par exemple, les commandes valides pour l'état d'authentification et de sélection sont listées dans les commandes d'état authentifié).

Les arguments de commande, identifié par "Arguments:" dans les descriptions au-dessous, sont décrites par fonction, et non par syntaxe. La syntaxe précise des arguments de commande est décrite dans la partie Syntaxe Formelle.

Certaines commandes font que le serveur doit retourner des réponses bien spécifiques. Elles sont identifiées par "Responses:" dans les descriptions de commandes qui suivent. Regardez les descriptions de réponses dans la section des réponses pour des informations sur celles-ci et la section des Syntaxes Formelles pour la syntaxe précise de ces réponses. Il est possible que les données serveurs soient transmises en tant que résultat à n'importe quelle commande. Ainsi, les commandes qui ne réclame précisément pas de données serveurs spécifie "pas de réponses spécifiques pour cette commande" ("no specific responses for this command") au lieu de "rien" ("none").

Le "Résultat:" dans la commande de description fait référence aux réponses possibles d'état marqué (tagged) à une commande, et à toute interprétation particulière à ces réponses d'état.

- 6.1. Commandes Client - tout état

Les commandes suivantes sont valables pour tous les états : CAPABILITY, NOOP, et LOGOUT.

- 6.1.1. Commande CAPABILITY

Arguments: Aucun

Réponses: NECCESSITE (REQUIRED) la réponse non marquée : CAPABILITY

Résultat:
OK - capacité terminée
BAD - commande inconnue ou arguments non valides

La commande CAPABILITY demande la liste des possibilités que le serveur supporte. Le serveur DOIT (MUST) envoyer une seule réponse CAPABILITY non marquée avec "IMAP4rev1" comme étant une des capacités listées avant la réponse (marquée) OK. Cette liste de capacité est indépendante des états de connexion ou de l'utilisateur. Il n'est donc pas nécessaire d'émettre plus d'une fois une commande CAPACITY par connexion.

Un nom de capacité qui commence par "AUTH=" indique que le serveur supporte ce mécanisme particulier d'authentification. De tels noms font partie, par définition, de cette spécification. Par exemple, les capacités d'autorisation pour l'identificateur expérimental "blurdybloop" devrait être

"AUTH=XBLURDYBLOOP" et non "XAUTH=BLURDYBLOOP" ou "XAUTH=XBLURDYBLOOP".

D'autres noms de capacité font référence à des extensions, révisions, ou amendements à cette spécification. Regardez la documentation de la repense CAPABILITY pour plus d'information. Aucune capacité au-delà du jeu de base IMAP4rev1 défini dans cette spécification n'est capable d'invoquer une capacité sans une action explicite du client.

Voir la section qui a pour titre "Commandes Client - Expérimentale/Annexe" pour des informations à propos de la configuration du site ou des capacités d'implémentations.

Exemple:
C: abcd CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4
S: abcd OK CAPABILITY completed

- 6.1.2. Commande NOOP

Arguments: Aucun

Réponses: Pas de réponses spécifiques

Résultat:
OK - noop fait
BAD - commande inconnue ou arguments non valides

La commande NOOP réussit toujours. Elle ne fait rien.

Puisque toute commande peut retourner l'état d'une mise à jours en tant que donnée non marquée, la commande NOOP peut être utilisée pour sonder périodiquement pour rechercher de nouvelles mises à jours de statut de message ou pour rechercher de nouveaux messages pendant une période d'inactivité. La commande NOOP peut aussi être utilisée pour remettre à zéro le timer autologout d'inactivité.

Exemple:
C: a002 NOOP
S: a002 OK NOOP completed
. . .
C: a047 NOOP
S: * 22 EXPUNGE
S: * 23 EXISTS
S: * 3 RECENT
S: * 14 FETCH (FLAGS (\Seen \Deleted))
S: a047 OK NOOP completed

- 6.1.3. Commande LOGOUT

Arguments: Aucun

Réponses: NECCESSITE (REQUIRED) une réponse non marquée : BYE

Résultat: OK - logout terminé
BAD - commande inconnue ou arguments non valides

La commande LOGOUT informe le serveur que le client en a fini avec la connexion. Le serveur DOIT (MUST) envoyer une réponse non marquée BYE avant la réponse marquée OK, et ensuite fermer la connexion réseau.

Exemple: C: A023 LOGOUT
S: * BYE IMAP4rev1 Server logging out
S: A023 OK LOGOUT completed

(le serveur et le client ferme alors la connexion)

- 6.2. Commandes client - Etat non authentifié

En état non-authentifié, la commande LOGIN ou AUTHENTICATE établissent l'authentification et entre dans un état authentifié.

La commande AUTHENTICATE fournie un mécanisme général pour une variété de technique d'authentification, puisque la commande LOGIN utilise traditionnellement le nom de l'utilisateur et le mot de passe, qui va de pair sous forme d'un texte non codé.

Les implémentations serveur PEUVENT (MAY) permettre l'accès non authentifié à des boîtes aux lettres. La convention est d'utiliser une commande LOGIN avec le userid "anonymous". Un mot de passe est réclamé (REQUIRED). C'est lié à l'implémentation de savoir qu'elles sont les exigences (s'il y en a) qui sont attaché au mot de passe et quels sont les restrictions d'accès qui sont lié aux utilisateurs en accès anonyme.

Une fois authentifié (y compris en anonyme), il n'est pas possible de reéntrer dans l'état non authentifié.

En plus des commandes universelles (CAPABILITY, NOOP, et LOGOUT), les commandes suivantes sont valables dans un état non authentifié : AUTHENTICATE et LOGIN.

- 6.2.1. Commande AUTHENTICATE

Arguments: Nom du mécanisme d'authentification

Réponses: Des données complémentaires peuvent être demandées

Résultat: OK - authentification faite, maintenant en état authentifié
NO - échec d authentification : mécanisme d'authentification non supporte, lettres de créances rejetées (non reconnue)
BAD - commande inconnue ou arguments non valides, échange d'authentification arrêté

La commande AUTHENTICATE indique au serveur un mécanisme d'authentification, tel qu'il est décrit dans [IMAP-AUTH]. Si le serveur supporte le mécanisme d'authentification demandé, il exécute un échange protocolaire d'authentification afin d'authentifier et identifier le client. Il PEUT (MAY) aussi passer à un mécanisme OPTIONNEL (OPTIONAL) de protection pour des interactions ultérieures de protocole. Si le mécanisme d'authentification demandé n'est pas supporté, le serveur DEVRAIT (SHOULD) rejeter la commande AUTHENTICATE en envoyant une réponse

NO marquée.

L'échange protocolaire d'authentification consiste en séries d'interpellation du serveur et de réponses du client qui sont spécifiques au mécanisme d'authentification. La demande du serveur comprends une commande de continuation de réponse avec le marqueur "+" suivi par une chaîne encodée en BASE64. Les réponses du client consiste en une ligne comprenant une chaîne encodée BASE64. Si le client veut arrêter un échange d'authentification, il fait parvenir une ligne avec un seul "*". Si le serveur reçoit une telle réponse, il DOIT (MUST) rejeter la commande AUTHENTICATE par l'envoi d'une réponse marqué BAD.

Un mécanisme de protection fournit une protection pour l'intégrité et la confidentialité à la connexion. Si un mécanisme de protection est initié, cela s'applique à toutes les données qui suivent et qui seront envoyés tout au long de la connexion ; Le mécanisme de protection prends effet immédiatement après le CRLF qui achève l'échange pour une authentification par le client, et le CRLF de la réponse marqué OK pour le serveur. Une fois que le mécanisme de protection prend effet, le flux d'octets de commande et de réponse est traité dans les buffers en texte chiffrée. Chaque buffer est transmis par la connexion comme un flux de bits préfixe avec un champ de 4 octets arrangé de la même manière que le réseau ordonne les octets, et qui représente la taille des données qui suivent. La taille maximum des buffers de texte crypté est définit par le mécanisme de protection.

Les mécanismes d'authentifications sont OPTIONNELS (OPTIONAL).Les mécanismes de protections sont aussi OPTIONNELS. Les mécanismes d'authentification peuvent (MAY) être implementé sans aucun mécanisme de protection. Si une commande AUTHENTICATE échoue sans la réponse NO, le client peut essayer un autre mécanisme d'authentification en émettant une autre commande, ou PEUT (MAY) essayer de s'authentifier en utilisant la commande LOGIN. En d'autre terme, le client PEUT (MAY) demander des types d'authentification en les déclinant par ordre de préférence, avec la commande LOGIN en dernier recours.

Exemple:
S: * OK KerberosV4 IMAP4rev1 Server
C: A001 AUTHENTICATE KERBEROS_V4
C: + AmFYig==
C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3
/IJmrMG+25a4DT+nZImJ
jnTNHJUtxAA+o0KPKfHEcAFs9a3CL5Oebe
/ydHJUwYFdWwuQ1MWiy6IesKvj
L5rL9WjXUb9MwT9bpObYLGOKi1Qh
S: + or//EoAADZI=
C: DiAF5A4gA+oOIALuBkAAmw==
S: A001 OK Kerberos V4 authentication successful

Remarque: Les lignes se coupent dans la première réponse du client pour des questions de clarté de présentation, en fait, elles ne le sont pas pour de véritables authentifiants.

- 6.2.2. Commande LOGIN

Arguments: Nom d'utilisateur
Mot de passe

Réponses: Pas de réponses spécifiques pour cette commande

Résultat:
- OK - login achevé, maintenant en état authentifié
- NO - login a échoué : nom d'utilisateur ou mot de passe rejeté
- BAD - commande inconnue ou arguments non valides

La commande LOGIN identifie les clients pour le serveur et porte le mot de passe sous forme d'un texte en clair qui identifie l'utilisateur.

Exemple:
C: a001 LOGIN SMITH SESAME
S: a001 OK LOGIN completed

- 6.3. Commandes clients - état authentifié

En état authentifié, les commandes qui manipulent les boîtes aux lettres en tant qu'entités atomiques sont permises. Hors de ces commandes, les commandes SELECT et EXAMINE sélectionneront une boite aux lettres pour l'accès et entrer dans l'état de sélection.

En plus des commandes universelles (CAPABILITY, NOOP, et LOGOUT), les commandes suivantes sont valides pour l'état authentifié : SELECT, EXAMINE, CREATE, DELETE, RENAME, SUBSCRIBE, UNSUBSCRIBE, LIST, LSUB, STATUS, et APPEND.

- 6.3.1. Commande SELECT

Arguments: Nom de boîtes aux lettres

Réponses: NECESSITE (REQUIRED) les réponses marquées : FLAGS, EXISTS, RECENT.
Les réponses OK non marquées et FACULTATIVES (OPTIONAL) sont: UNSEEN, PERMANENTFLAGS

Résultat:
- OK - sélection effectuée, maintenant en mode sélectionnée
- NO - échec de sélection, maintenant en mode authentifié: pas de boîte aux lettres, ne peut accéder à la boîte aux lettres
- BAD - commande inconnue ou arguments non valides

La commande SELECT sélectionne une boîte aux lettres, ainsi les messages dans la boîte aux lettres sont accessibles. Avant de retourner OK au client, le serveur DOIT (MUST) envoyer les données suivantes non marquées au client :

FLAGS	Drapeaux définis dans la boîte aux lettres. Voir la description de la réponse FLAG pour plus de détail.
<n> EXISTS	Le nombre de messages dans la boîte aux lettres. Voir la description de la réponse EXISTS pour plus de détail.
<n> RECENT	Le nombre de messages avec le drapeau \Recent mis. Voir la description de la réponse RECENT pour plus de détail.
OK [UIDVALIDITY <n>]	La valeur d'identification d'unique de validité. Voir la description de la commande UID pour plus de détail.

pour définir l'état initial d'une boite aux lettres d'un client.

Le serveur DEVRAIT (SHOULD) aussi envoyer un code de réponse UNSEEN dans une réponse OK non marquée, en indiquant le numéro de séquence du premier message non vu dans la boîte aux lettres.

Si le client n'a pas la possibilité de changer l'état permanent d'un ou plusieurs états des marqueurs listés dans la réponse non marquée FLAGS, le serveur doit (should) envoyer un code de réponse PERMANENTFLAGS dans la réponse OK non marquée, en donnant la liste les drapeaux que le client peut changer de façon permanente.

Seule une boîte aux lettres peut être sélectionnée à un moment donnée dans une connexion ; l'accès simultané à de multiple boîte aux lettres requiers de multiples connexions. La commande SELECT désélectionne automatiquement toute boîte aux lettres actuellement sélectionné avant de tenter une nouvelle sélection. Par conséquent, si une boîte aux lettres est sélectionnée et que la commande SELECT est tentée et échoue, aucune boite aux lettres n'est sélectionnée.

Si le client a l'autorisation de modifier la boîte aux lettres, le serveur DEVRA (SHOULD) préfixer le texte de la réponse OK marqué avec le code de réponse "[READ-WRITE]".

Si le client n'a pas la permission de modifier la boîte aux lettres mais à l'autorisation d'accès en lecture seule, la boîte aux lettres est sélectionnée en lecture seule, et le serveur DOIT (MUST) préfixer le texte de sa réponse de OK marquée par le code de réponse "[READ-ONLY]. L'accès en réponse lecture-seule (read-only) au travers de la commande SELECT diffère de la commande EXAMINE du fait que l'accès bien établi en lecture de certaines boîtes aux lettres PEUT (MAY) permettre le changement d'état permanent par utilisateur (que l'on oppose de global). Les messages de netnews marqué en tant que fichier.newsrc sur un serveur est un exemple en tant qu'état permanent par utilisateur pouvant être modifié pour des boîtes aux lettres en lecture seule.

Exemple:
C: A142 SELECT INBOX
S: * 172 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Message 12 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen *)] Limited
S: A142 OK [READ-WRITE] SELECT completed

- 6.3.2. Commande EXAMINE

Arguments: Nom de boîte aux lettres

Réponses: NECCESSITE (REQUIRED) les réponses non marquées : FLAGS, EXISTS, RECENT.
Les réponses OK de type non marquées, et OPTIONNELLE (OPTIONAL) sont : UNSEEN, PERMANENTFLAGS

Résultat:
OK - vérification faite, et maintenant en mode sélectionné
NO - erreur dans la vérification, maintenant en état authentifié, pas de telle boîte aux lettres, ne peut accéder à la boîte aux lettres

BAD - commande inconnue ou arguments non valides

La commande EXAMINE est identique à la commande SELECT et retourne les mêmes sorties ; Cependant, la boîte sélectionnée est identifiée en lecture seulement. Aucun changement dans l'état permanent de la boîte aux lettres, y compris le changement d'état par utilisateur, n'est permis.

Le texte d'une réponse marqué OK à la commande EXAMINE DOIT (MUST) commencer avec le code réponse "[READ-ONLY]".

Exemple:
C: A932 EXAMINE blurdybloop
S: * 17 EXISTS
S: * 2 RECENT
S: * OK [UNSEEN 8] Message 8 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS ()] No permanent flags permitted
S: A932 OK [READ-ONLY] EXAMINE completed

- 6.3.3. Commande CREATE

Arguments: Nom de boîte aux lettres

Réponses: Pas de réponse spécifique pour cette commande

Résultat:
OK - création effectuée
NO - échec de création : ne peut créer la boîte aux lettres avec ce nom
BAD - commande inconnue ou argument non valide

La commande CREATE crée une boîte aux lettres avec un nom donné. Une réponse OK est retournée seulement si une nouvelle boîte aux lettres avec ce nom a été créée. C'est une erreur de s'attendre a créer INBOX, ou une boîte aux lettres avec un nom qui existe déjà. Toute erreur dans la création donnera lieu a une réponse NO marquée.

Si le nom de la boîte aux lettres est suffixé par le caractère séparateur de hiérarchique du serveur (comme celui qui est retourné par la commande LIST), c'est une déclaration selon laquelle le client a l'intention de créer des noms de boîtes aux lettres sous ce nom dans la hiérarchie. Les implémentations serveur qui n'ont pas besoin de cette déclaration DOIVENT (MUST) l'ignorer.

Si le caractère séparateur hiérarchique du serveur apparaît autre part dans le nom, le serveur DEVRAIT (SHOULD) créer tous noms hiérarchiquement supérieurs qui sont nécessaires pour la commande CREATE se finissent correctement. En d'autres termes, l'intention de créer "foo/bar/zap" sur un serveur pour lequel "/" est le caractère de séparation hiérarchique DEVRAIT (SHOULD) créer foo/ et foo/bar s'ils n'existaient pas.

Si une nouvelle boîte aux lettres est crée avec le même nom qu'une boîte aux lettres qui n'avait pas été supprimé, ses identifiants uniques utilisés DOIVENT (MUST) être plus grand que n'importe quels autres unique identifiants utilisés dans la précédente incarnation de la boîte aux lettres à moins que la nouvelle incarnation est eue une valeur de validé d'identifiant unique différante.

Voir la description d'une commande UID pour plus de détail.

Exemple: C: A003 CREATE owatagusiam/
S: A003 OK CREATE completed
C: A004 CREATE owatagusiam/blurdybloop
S: A004 OK CREATE completed

Remarque: L'interprétation de cet exemple dépend du fait que le "/" est retourné en tant que séparateur hiérarchique de LIST. Si "/" est le séparateur hiérarchique, un nouveau niveau de la hiérarchie nommé "owatgusiam" avec un membre appelé "blurdybloop" est crée. Sinon, deux boîtes aux lettres au même niveau de hiérarchie sont crées.

- 6.3.4. Commande DELETE

Arguments: Nom de boîte aux lettres

Réponses: Pas de réponse spécifique pour cette commande

Résultat:
OK - suppression effectuée
NO - échec de suppression : On ne peut pas supprimer une boîte aux lettres avec ce nom
BAD - commande inconnue ou arguments non valides

La commande DELETE enlevé de façon permanente la boite aux lettres qui a le nom que l'on a donné. Une réponse OK marquée est retournée seulement si la boîte aux lettres a été supprimée. C'est une erreur que de s'attendre à supprimer INBOX ou un nom de boîte aux lettres qui n'existe pas.

La commande DELETE NE DOIT PAS (MUST NOT) supprimer les noms hiérarchiquement inférieurs. Par exemple, si une boîte aux lettres "foo" a un inférieur "foo.bar" (en supposant que "." est le caractère de délimitation hiérarchique), le fait d'enlever "foo" NE DOIT PAS (MUST NOT) enlever "foo.bar". C'est une erreur que de s'attendre à supprimer un nom qui a des noms de hiérarchie inférieure et aussi qui a l'attribut de nom de boîte aux lettres \Noselect (voir la description de la réponse LIST pour plus de détails).

Il est permis de supprimer un nom qui a des noms de hiérarchie inférieure et qui n'a pas l'attribut de nom de boîtes aux lettres \Noselect. Dans ce cas, tous les messages dans cette boîte aux lettres sont enlevés et le nom obtiendra l'attribut de nom de boite aux lettres \Noselect.

La valeur de l'identifiant unique de l'utilisateur de plus haut niveau d'une boîte aux lettres supprimé DOIT (MUST) être préservé de façon à ce que une nouvelle boite aux lettres crée avec le même nom ne réutilisera pas les identifiants de la précédente incarnation, A MOINS QUE la nouvelle incarnation ait une valeur de validité d'identifiant unique différente.

Voir la description de la commande UID pour plus de détail.

Exemples: C: A682 LIST "" *
S: * LIST () "/" blurdybloop
S: * LIST (\Noselect) "/" foo
S: * LIST () "/" foo/bar
S: A682 OK LIST completed
C: A683 DELETE blurdybloop

S: A683 OK DELETE completed
C: A684 DELETE foo
S: A684 NO Name "foo" has inferior hierarchical names
C: A685 DELETE foo/bar
S: A685 OK DELETE Completed
C: A686 LIST "" *
S: * LIST (\Noselect) "/" foo
S: A686 OK LIST completed
C: A687 DELETE foo
S: A687 OK DELETE Completed

C: A82 LIST "" *
S: * LIST () "." blurdybloop
S: * LIST () "." foo
S: * LIST () "." foo.bar
S: A82 OK LIST completed
S: A83 DELETE blurdybloop
S: A83 OK DELETE completed
C: A84 DELETE foo
S: A84 OK DELETE Completed
C: A85 LIST "" *
S: * LIST () "." foo.bar
S: A85 OK LIST completed
C: A86 LIST "" %
S: * LIST (\Noselect) "." foo
S: A86 OK LIST completed

- 6.3.5. Commande RENAME

Arguments: Nom de boîte aux lettres qui existe déjà
Nouveau nom de boîte aux lettres
Réponses: Pas de réponses spécifiques pour cette commande

Résultat:
OK - renommange effectué
NO - échec de renommage : ne peut renommer la boîte aux lettres qui a ce nom, ne peut renommer avec ce nom de boîte aux lettres.
BAD - commande inconnue ou argument non valide

La commande RENAME change le nom d'une boîte aux lettres. Une réponse de marqueur OK est retournée seulement si la boîte aux lettres a été renommée. C'est une erreur que de s'attendre à renommer une boite qui n'existe pas ou de renommer en un nom de boîte aux lettres qui existe déjà. Toute erreur dans le renommage aura pour effet de retourner une réponse NO marquée.

Si le nom avait des noms de hiérarchie inférieure, alors les noms hiérarchiquement inférieurs DOIVENT (MUST) aussi être renommé.

Par exemple, un renommage de "foo" en zap renommera "foo/bar" (en supposant que "/" est le caractère de délimitation hiérarchique) en "zap/bar".

La valeur de l'identifiant unique le plus grand utilisé d'un nom de vieille boîte aux lettres DOIT (MUST) être préservé pour que si une nouvelle boîte aux lettres est crée avec le même nom, on n'utilisera pas les identifiants dans une incarnation ultérieure, A MOINS, que la nouvelle incarnation ait une valeur de validité d'identification unique différente. Voir la description de la commande UID pour plus de détail.

Le renommage de INBOX est autorisé, et a un comportement spécial. Elle déplace tous les messages contenus dans INBOX vers une nouvelle boîte aux lettres portant le nom donné, laissant INBOX vide. Si l'implémentation d'un serveur supporte des noms de hiérarchie inférieurs à INBOX, ceux-ci ne sont pas affectés par le renommage de INBOX.

Exemples:
C: A682 LIST "" *
S: * LIST () "/" blurdybloop
S: * LIST (\Noselect) "/" foo
S: * LIST () "/" foo/bar
S: A682 OK LIST completed
C: A683 RENAME blurdybloop sarasoop
S: A683 OK RENAME completed
C: A684 RENAME foo zowie
S: A684 OK RENAME Completed
C: A685 LIST "" *
S: * LIST () "/" sarasoop
S: * LIST (\Noselect) "/" zowie
S: * LIST () "/" zowie/bar
S: A685 OK LIST completed

C: Z432 LIST "" *
S: * LIST () "." INBOX
S: * LIST () "." INBOX.bar
S: Z432 OK LIST completed
C: Z433 RENAME INBOX old-mail
S: Z433 OK RENAME completed
C: Z434 LIST "" *
S: * LIST () "." INBOX
S: * LIST () "." INBOX.bar
S: * LIST () "." old-mail
S: S: Z434 OK LIST completed

- 6.3.6. Commande SUBSCRIBE

Arguments: Boîte aux lettres

Réponses: Pas de réponse spécifique pour cette commande

Résultat: OK - souscription effectuée (abonnement fait)
NO - échec de souscription : ne peut souscrire avec ce nom
BAD - commande inconnue ou arguments non valides

La commande SUBSCRIBE ajoute le nom de boîte aux lettres spécifiées au jeu de boîte aux lettres "active" ou subscribed" du serveur de la même façon que ce qui est retourné par la commande LSUB. Cette commande retourne la réponse ok marquée seulement si la souscription est un succès.

Un serveur PEUT (MAY) valider l'argument de la boîte aux lettres par SUBSCRIBE pour vérifier qu'elle existe. Cependant, il NE DOIT PAS (MUST NOT) enlever unilatéralement un nom de boîte aux lettres qui existe de la liste de souscription même si la boîte aux lettres par ce nom n'existe plus.

Remarque: Ceci est requis parce que certains sites de serveur peuvent systématiquement enlever une boîte aux lettres avec un nom de connu (par exemple "alertes-systemes") après que son contenu ait expiré, avec l'intention de la recréer avec un nouveau contenu quand c'est approprié.

Exemple: C: A002 SUBSCRIBE #news.comp.mail.mime
S: A002 OK SUBSCRIBE completed

- 6.3.7. Commande UNSUBSCRIBE

Arguments: Nom de boîte

Réponses: Pas de réponse spécifique pour cette boîte aux lettres

Résultat: OK - désabonnement effectué
NO - unsubscribe a échoué : ne peut se désabonner de ce nom
BAD - commande inconnue ou argument non valide

La commande UNSUBSCRIBE enlève le nom de boîte aux lettres spécifié au jeu de boîte aux lettres "active" ou "subscribed" du serveur pareil à ce qui est retourné par la commande LSUB. Cette commande retourne la réponse OK marquée seulement si la désouscription est un succès.

Exemple: C: A002 UNSUBSCRIBE #news.comp.mail.mime
S: A002 OK UNSUBSCRIBE completed

- 6.3.8. Commande LIST

Arguments: Nom de référence
Nom de boîte aux lettres avec de possible joker

Réponses: Réponses non marquées : LIST

Résultat: OK - liste effectuée
NO - échec à la création de liste : ne peut lister cette référence ou ce nom
BAD - commande inconnue ou argument non valide

La commande LIST retourne un sous jeu de nom du jeu complet des noms disponibles pour le client. Aucunes ou plusieurs réponses de LIST non marquées sont retournées, contenant les attribues de nom, les délimiteurs de hiérarchie, et le nom ; Voir la description de la réponse LIST pour plus de détail.

IL FAUT (SHOULD) que la commande LIST retourne ses données rapidement, sans délais indus. Par exemple, on NE DEVRA PAS (SHOULD NOT) générer trop de problème pour le calcul du statut \Marked ou \Unmarked ou pour exécuter d'autre traitement ; Si chaque nom nécessite 1 seconde de traitement, alors une liste de 1200 noms prendra 20 minutes !

Un argument de nom de référence vide (chaîne "")indique que le nom de la boîte aux lettres est interprété comme si c'était fait par SELECT. Les noms de boîtes aux lettres retournées DOIVENT (MUST) correspondre au modèle de nom de boîtes aux lettres fournies. Un argument de nom de référence non vide est le nom de la boîte aux lettres ou un niveau de hiérarchie de boîtes aux lettres, et indique le contexte dans lequel le nom de la boîte aux lettres est interprété à la manière défini par l'implémentation.

Un argument de nom de boîte aux lettres vides (chaîne "") constitue une demande spéciale qui permet de retourner le délimiteur de hiérarchie et le nom qui est en racine (root) du nom donné dans la référence. La valeur retournée en tant que racine PEUT (MAY) être nulle si la référence n'est pas racine ou est null. Dans tous les cas, le délimiteur hiérarchique est retourné. Cela permet au client d'obtenir le délimitateur hiérarchique même quand il n'y a aucune boîte aux lettres portant actuellement ce nom.

La référence et les arguments de nom de boîte aux lettres sont interprétés, d'une manière dépendante de l'implémentation, en une forme canonique qui représente une hiérarchie de gauche à droite non ambigu. Les noms de boîtes à lettres retournés sont retourné dans la forme interprétée.

Toutes parties de l'argument de la référence qui est incluse dans la forme interprétée DEVRAIT (SHOULD) préfixer la forme interprétée. Ce DOIT (SHOULD) aussi être sous la même forme que l'argument du nom de référence. Cette règle permet aux clients de déterminer si le nom de la boîte aux lettres retournée est dans le contexte d'un argument de référence, ou si quelque chose à propos de l'argument de la boîte aux lettres a plus d'importance (passer outre) que l'argument de référence. Sans cette règle, le client aurait a connaître les sémantiques de nommage du serveur y compris les caractères qui "s'échappent" dans le contexte de nommage.

Par exemple, voici des exemples où comment les références et les noms de boîtes aux lettres pourraient être interprété sur un serveur basé sur UNIX :

Référence	Nom de boîte aux lettres	Interprétation
~smith/Mail/	foo.*	~smith/Mail/foo.*
archive/	%	archive/%
#news.	comp.mail.*	#news.comp.mail.*
~smith/Mail/	/usr/doc/foo	/usr/doc/foo
archive/	~fred/Mail/*	~fred/Mail/*

Les trois premiers exemples démontre les interprétations dans le contexte d'argument de référence ; Remarquez que "~smith/Mail" NE DEVRAIT PAS (SHOULD NOT) être transformé en quelque chose comme "/u2/users/smith/Mail", ou bien ce serait impossible pour le client de déterminer ce que l'interprétation était dans le contexte de la référence.

Le caractère "*" est un caractère joker, et il correspond à zéro ou plusieurs caractère à cet endroit. Le caractère "%" est similaire a "*" mais ne correspond pas à un délimiteur hiérarchique. Si le joker "%" est le dernier caractère d'un argument de nom de boîte aux lettres, correspondant aux niveaux

de hiérarchie, alors ces niveaux sont aussi retournés. Si ces niveaux de hiérarchie ne sont pas aussi des boîtes aux lettres sélectionnable, ils sont retournés avec l'attribut de nom de boîte aux lettres \Noselect (voir la description de la réponse LIST pour plus de détails).

Les implémentations des serveurs ont l'autorisation de "cacher" de manière différente les boîtes aux lettres accessibles par les caractères jokers, de façon à empêcher certains caractères ou nom d'entrer en correspondance avec un joker pour des situations bien établie. Par exemple, un serveur basé sur UNIX pourrait restreindre l'interprétation de "*" de façon à ce que le caractère "/" en première position ne corresponde rien.

Le nom particulier INBOX est compris dans les sorties de LIST, si INBOX est supporté par ce serveur pour ce user et si la chaîne en majuscule "INBOX" correspond à la référence interprétée et aux arguments de nom de boîte aux lettres avec les caractères joker comme décrit au-dessus. Le critère pour l'omission de INBOX est si SELECT INBOX retourne une erreur ; On considère que ce n'est pas pertinent si le véritable utilisateur de INBOX réside sur ce serveur ou sur un autre.

Exemple:
C: A101 LIST "" ""
S: * LIST (\Noselect) "/" ""
S: A101 OK LIST Completed
C: A102 LIST #news.comp.mail.misc ""
S: * LIST (\Noselect) "." #news.
S: A102 OK LIST Completed
C: A103 LIST /usr/staff/jones ""
S: * LIST (\Noselect) "/" /
S: A103 OK LIST Completed
C: A202 LIST ~/Mail/ %
S: * LIST (\Noselect) "/" ~/Mail/foo
S: * LIST () "/" ~/Mail/meetings
S: A202 OK LIST completed

- 6.3.9. Commande LSUB

Arguments: Nom de référence
Nom de référence avec de possibles caractères jokers

Réponses: Réponses non marques : LSUB

Résultat:
OK - LSUB effectué
NO - LSUB a échoué ; ne peut lister cette référence ou ce nom
BAD - Commande inconnue ou arguments non valides

La commande LSUB retourne un sous jeu de nom du lot de nom que l'utilisateur a déclaré comme étant "active" (actif) ou souscribed" (Abonné). Plusieurs ou aucunes réponses LSUB, de type non marqué, peuvent être retournées. Les arguments de LSUB sont de la même forme que ceux pour LIST.

Un serveur PEUT (MAY) valider les noms souscrits pour voir s'ils existent toujours. Si le nom n'existe pas, Il doit être marqué avec l'attribut \Noselect dans la réponse à LSUB. Le serveur NE DOIT PAS (MUST NOT) unilatéralement enlever un nom de boîte aux lettres existant de la liste

souscrite même si une boîte aux lettres sous ce nom n'existe plus.

Exemple: C: A002 LSUB "#news." "comp.mail.*"
S: * LSUB () "." #news.comp.mail.mime
S: * LSUB () "." #news.comp.mail.misc
S: A002 OK LSUB completed

- 6.3.10. Commande STATUS

Arguments: Nom de boîte aux lettres
Noms de statut

Réponses: Réponses non marquées : STATUS

Résultat: OK - status completed
NO - status failure : Pas de statut pour ce nom
BAD - commande inconnue ou arguments non valides

La commande STATUS demande le statut de la boîte aux lettres indiquées. Cela ne modifie pas la boîte aux lettres qui est actuellement sélectionné, ni n'affecte l'état des messages situé dans la boîte aux lettres où s'effectue la demande (en particulier, STATUS NE DOIT PAS (MUST NOT) à l'origine de la perte pour des messages du drapeau \Recent).

La commande STATUS fournit une alternative à l'ouverture d'une seconde connexion IMAPrev1 et à l'action de passer la commande EXAMINE sur une boîte aux lettres pour demander le statut de cette dernière sans déselectionner l'actuelle boite aux lettres qui utilise la première connexion IMAP4rev1.

A la différence de la commande LIST, la commande STATUS n'est pas garantie qu'a la rapidité de ses réponses. Pour certaines implémentations, le serveur est obligé d'ouvrir la boîte aux lettres en interne en lecture-seulement pour obtenir une information de statut sûre. De plus, à la différence de la commande LIST, la commande STATUS n'accepte pas les jokers.

Les éléments de donnée de statut défini actuellement que l'on peut demander sont :

MESSAGES Le nombre de messages dans la boite aux lettres.

RECENT Le nombre de messages avec le marqueur à \Recent.

UIDNEXT La prochaine valeur d'UID qui sera affecté à un nouveau message dans la boîte aux lettres. On a la garantie que cette valeur ne changera pas à moins que de nouveau messages soient ajoutés à cette boîte aux lettres, ou que cela changera quand de nouveaux messages seront ajoutés même si ces nouveaux messages sont effacés par la suite (expurged).

UIDVALIDITY La valeur de validité de l'identifiant unique de la boîte aux lettres.

UNSEEN Le nombre de messages qui ne sont pas avec le drapeau à /seen.

Exemple: C: A042 STATUS blurdybloop (UIDNEXT MESSAGES)
S: * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)
S: A042 OK STATUS completed

- 6.3.11. Commande APPEND

Arguments: Nom de boîte aux lettres
Liste FACULTATIVE (OPTIONAL) de marqueurs mis entre parenthèse
Chaîne FACULTATIVE (OPTIONAL) date/heure
Message littéral

Réponses: Pas de réponses spécifiques pour cette commande

Résultat: OK - append effectué
NO - erreur dans l'append : ne peut mettre au bout de cette boîte aux lettres, erreur dans les marqueurs, ou dans date/heure ou dans le message texte
BAD - commande inconnue ou arguments non valides

La commande APPEND ajoute l'argument littéral en tant que nouveau message à la fin de la boite aux lettres de destination spécifiée. Son argument DOIT (SHOULD) être au format de message [RFC-822]. Les caractères sur 8 bits sont autorisés dans le message. Une implémentation de serveur qui n'est pas capable de préserver les données sur 8 bits DOIT (MUST) être capable d'assurer la réversibilité en 8 bits des données ajoutées en 7 bits, qui ont été converti en utilisant l'encodage de transfert de contenu [MIME-IMB].

Remarque: Il PEUT (MAY) y avoir des exceptions, par exemple les brouillons de messages, pour lesquels les lignes d'en-tête de type [RFC-822] sont omises dans l'argument de message littéral pour lequel on fait l'APPEND. En faisant cela, l'ensemble des implications que cela sous-entends DOIT (MUST) être compris et être soupesé avec précaution.

Si une liste mise entre parenthèse de drapeau est spécifiée, les drapeaux DOIVENT (SHOULD) être mis sur le message qui en résulte ; Sinon, la liste de drapeau du message qui en résulte, est mise, par défaut, à vide.

Si une date_heure est précisée, la date interne DEVRAIT (SHOULD) être mise pour le message qui en résulte; Sinon la date interne du message qui en résulte est établie sur la date et l'heure par défaut.

Si l'ajout à la suite n'est pas un succès pour quelque raison que ce soit, la boîte aux lettres DOIT (MUST) être restauré en l'état où elle était avant que l'on désire faire l'append. Aucun ajout partiel n'est permis.

Si la boîte aux lettres de destination n'existe pas, un serveur DOIT (MUST) retourner une erreur, et ne DOIT PAS (MUST NOT) créer automatiquement la boîte aux lettres. A moins que l'on soit sûr de ne pas pouvoir créer la boîte aux lettres de destination, le serveur DOIT (MUST) envoyer le code de réponse "[TRYCREATE]" en tant que préfixe du texte de réponse NO non marquée. Cela donne l'indication au client qu'il peut essayer une commande de CREATE et de réessayer l'APPEND Si le CREATE est un succès.

Si la boîte aux lettres est actuellement sélectionné, les actions normales pour les nouveaux courriers

DEVRAIENT (SHOULD) s'effectuer. Plus précisément, le serveur DEVRAIT (SHOULD) avertir immédiatement le client par l'intermédiaire d'une réponse EXISTS de type non marquée. Si le serveur ne fait pas cela, le client PEUT (MAY) essayer une commande NOOP (ou à défaut, une commande CHECK) après une ou plusieurs commandes APPEND.

Exemple:
C: A003 APPEND saved-messages (\Seen) {310}
C: Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)
C: From: Fred Foobar <foobar@Blurdybloop.COM>
C: Subject: afternoon meeting
C: To: mooch@owatagu.siam.edu
C: Message-Id: <B27397-0100000@Blurdybloop.COM>
C: MIME-Version: 1.0
C: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
C:
C: Hello Joe, do you think we can meet at 3:30 tomorrow?
C:
C: A003 OK APPEND completed

Remarque: La commande APPEND n'est pas utilisée pour la distribution de message, du fait qu'elle ne fournit pas de mécanisme pour transférer les informations d'enveloppe [SMTP].

- 6.4. Commandes clients- Etat sélectionné

En état sélectionné, les commandes qui manipulent les messages dans une boîte aux lettres sont autorisées.

En plus des commandes universelles (CAPABILITY, NOOP, and LOGOUT), et des commandes d'état authentifié (SELECT, EXAMINE, CREATE, DELETE, RENAME, SUBSCRIBE, UNSUBSCRIBE, LIST, LSUB, STATUS, et APPEND), les commandes suivantes sont valides pour l'état sélectionné : CHECK, CLOSE, EXPUNGE, SEARCH, FETCH, STORE, COPY, et UID.

- 6.4.1. Commande CHECK

Arguments: Aucuns

Réponses: Pas de réponses spécifiques pour cette commande

Résultat:
OK - contrôle effectué
BAD - commande inconnue ou arguments non valides

La commande CHECK demande un point de contrôle sur la boîte aux lettres actuellement sélectionnée. Un point de contrôle fait référence à tout soins dépendant de l'implémentation associés à la boite aux lettres (par exemple, mettre en adéquation l'état de la mémoire du serveur pour la boîte aux lettres avec celle du disque) et qui n'est pas exécuter normalement en tant que partie de chacune des commandes. Un point de contrôle PEUT (MAY) prendre une certaine quantité de temps pour s'exécuter. Si une implémentation de serveur n'a pas de tels soucis de sauvegarde, CHECK est équivalent à NOOP. Il n'y aucune garantie qu'une réponse non marquée EXISTS

arrivera comme résultat de CHECK. NOOP, et pas CHECK, DEVRAIT (SHOULD) être utilisé pour un nouveau sondage (polling) de boîte aux lettres.

Exemple: C: FXXZ OK CHECK Completed
S: FXXZ OK CHECK Completed

- 6.4.2. Commande CLOSE

Arguments: Aucuns

Réponses: Pas de réponses spécifiques pour cette commande

Résultat: OK - close effectué,maintenant en état authentifié
BAD - commande inconnue ou arguments non valides

La commande CLOSE enlève de façon définitive, de la boîte aux lettres actuellement sélectionnées, tous les messages qui ont le drapeau établi à \Deleted, et retourne de l'état sélectionné à l'état authentifié. Aucune réponse d'EXPUNGE, de type non marqué, est envoyée.

Aucun message n'est enlevé, et aucune erreur n'est renvoyée, si la boîte aux lettres a été sélectionnée par une commande EXAMINE ou si elle a été sélectionnée en lecture seule.

Même si une boîte aux lettres est sélectionnée, une commande de SELECT, EXAMINE ou LOGOUT PEUT (MAY) aboutir sans qu'il y ait eu au préalable l'aboutissement de la commande CLOSE. Les commandes SELECT, EXAMINE, et LOGOUT ferme implicitement la boîte aux lettres sélectionnée sans faire un EXPUNGE. Cependant quand beaucoup de messages sont supprimés, une séquence de CLOSE-LOGOUT ou CLOSE-SELECT est considérablement plus rapide qu'un EXPUNGE-LOGOUT ou qu'un EXPUNGE-SELECT du fait que les réponses EXPUNGE non marquées (que le client ignorera probablement) sont envoyées.

Exemple: C: A341 CLOSE
S: A341 OK CLOSE completed

- 6.4.3. Commande EXPUNGE

Arguments: Aucun

Réponses: Réponses de type non marquée : EXPUNGE

Résultat: OK - expunge effectué
NO - expunge a échoué : ne peut effacer, rayer, supprimer (par exemple permission refusée)
BAD - commande inconnue ou arguments non valides

La commande EXPUNGE enlève de façon définitive de la boîte aux lettres actuellement sélectionnée tous les messages qui ont le marqueur établi à \Delete. Avant de retourner OK au client, une réponse non marquée EXPUNGE est envoyer pour chaque message enlevé.

Exemple: C: A202 EXPUNGE

S: * 3 EXPUNGE
S: * 3 EXPUNGE
S: * 5 EXPUNGE
S: * 8 EXPUNGE
S: A202 OK EXPUNGE completed

Remarque: Dans cet exemple, les messages 3, 4, 7, et 11 avait le drapeau mis à \Deleted. Voir la description de la réponse EXPUNGE pour de plus ample explication.

- 6.4.4. Commande SEARCH

Arguments: Spécification OPTIONNELLE (OPTIONAL) du jeu de caractère [CHARSET]
Critère de recherche (un ou plusieurs)

Réponses: Réponses: Réponse REQUISE (REQUIRED) de type non marqué : SEARCH

Résultat:
OK - recherche effectuée
NO - erreur de recherche : ne peut chercher cette série de caractère [CHARSET] ou ce critère
BAD - commande inconnue ou arguments non valides

La commande SEARCH cherche dans la boîte aux lettres des messages qui correspondent aux critères de recherches données. Les critères de recherche sont composés par en une ou plusieurs clef de recherche. La réponse SEARCH non marquée du serveur contient la liste des numéros de séquence de message correspondant aux messages qui sont concordance avec dans le critère de la recherche.

Quand de multiples clés sont spécifiées, le résultat est l'intersection (fonction ET) de tous les messages qui sont en correspondance avec ces clefs. Par exemple, le critère DELETED FROM "SMITH" SINCE 1-Feb-1994 fait référence à tous les messages supprimés de Smith qui ont été place dans la boîte aux lettres depuis le 1 février 1994. Une clef de recherche peut être aussi une liste mise entre parenthèses composée de une ou plusieurs clef de recherche (par exemple pour l'utilisation avec les clés OR et NOT).

Les implémentations de serveurs PEUVENT (MAY) exclure des parties de corps [MIME-IMB] avec des types de support de contenu finaux autres que TEXT et MESSAGE, des conditions de correspondance de recherche.

Les spécifications OPTIONNELLES [OPTIONAL] de série de caractère consistent a placer le mot "CHARSET" suivi par un le jeu de caractère référencé [CHARSET] de référence. Cela indique le jeu de caractère [CHARSET] utilisé danse la chaîne qui apparaît dans le critère de recherche. Le contenu [MIME-IMB] transféré sous la forme encodée, et les chaînes [MIME-HDRS] dans les en-têtes [RFC-822]/[MIME-IMB], DOIVENT (MUST) être décodé avant de comparer le texte dans le jeu de caractère [CHARSET] sauf ce qui est en US-ASCII. US-ASCII DOIT (MUST) être supporté ; D'autres jeux de caractères [CHARSET] PEUVENT (MAY) être supporte. Si le serveur ne supporte pas le jeu de caractère [CHARSET], on DOIT (MUST) retourner une réponse NO marquée (et pas BAD).

Pour toutes les clés de recherche qui utilise des chaînes, un message correspond à la clé si la chaîne est une sous chaîne de champ. La correspondance est non sensible à la casse.

Les clés de recherche définies sont celles qui suivent. Référez-vous la section sur la syntaxe formelle pour les définitions syntaxiques précises des arguments.

<message set>	Messages avec des numéros de séquence de messages correspondant à la série de numéro de séquence de messages spécifié.
ALL	Tous les messages dans la boîte aux lettres. La clé initiale par défaut ainsi que pour le fait de faire des ET (for ANDing).
ANSWERED	messages avec le jeu de drapeaux \Answered.
BCC <string>	Messages qui contiennent la chaîne de caractère spécifié dans le champ BBC de la structure d'enveloppe.
BEFORE <date>	Messages dont la date interne est plus récente que la date spécifiée.
BODY <string>	Messages qui contiennent la chaîne spécifiée dans le corps du message
CC <string>	Messages qui contiennent la chaîne dans le champ CC de la structure d'enveloppe.
DELETED	Messages avec le drapeau \Deleted.
DRAFT	Messages avec le drapeau \Draft.
FLAGGED	Messages avec le drapeau \Flagged.
FROM <string>	Messages qui contiennent la chaîne dans le champ FROM de la structure d'enveloppe.
HEADER <field-name> <string>	Messages qui ont l'en-tête avec le nom de champs spécifié (comme ce qui est définie dans la [RFC-822]) et qui contienne la chaîne spécifiée dans le corps du champ de type [RFC-822].
KEYWORD <flag>	Messages avec le mot clé spécifié correspondants.
LARGER <n>	Messages avec une plus grande taille [RFC-822] que celle qui est précisé en nombre d'octets.
NEW	Messages qui ont les drapeaux \Recent mais qui n'ont pas le drapeau \Seen. C'est fonctionnellement équivalent à "(RECENT UNSEEN)".
NOT <clé de recherche>	Messages qui ne correspondent pas à la clé de recherche.

OLD	Messages qui n'ont pas le drapeau \Recent. C'est fonctionnellement équivalent à "NOT RECENT" (a mettre en contraste avec "NOT NEW" (sic)).
ON <date>	Messages qui ont leur date interne à la date spécifiée.
OR <clé de recherche1> <clé de recherche2>	Messages qui correspondent à l'une des clés de recherche.
RECENT	Messages avec le drapeau \Recent.
SEEN	Messages avec le drapeau \Seen.
SENTBEFORE <date>	Messages dont la date d'en-tête [RFC-822] est plus récente que la date spécifiée.
SENTON <date>	Messages dont la date d'en-tête [RFC-822] est égale à la date spécifiée
SENTSINCE <date>	Messages qui ont leur date interne égale ou plus tardive que la date spécifiée
SINCE <date>	Messages qui ont leur date interne égale ou plus tardive que la date spécifiée.
SMALLER <n>	Messages dont la taille (de type [RFC-822]) est plus petite que celle qui est spécifié en octet.
SUBJECT <string>	Messages qui contiennent la chaîne spécifiée dans le champ SUBJECT de la structure d'enveloppe.
TEXT <string>	Messages qui contiennent la chaîne spécifiée dans l'en-tête ou le corps du message.
TO <string>	Messages qui contiennent la chaîne spécifiée dans le champ TO de la structure d'enveloppe.
UID <message set>	Messages qui ont les identifiants uniques qui correspondent avec l'identifiant unique qui spécifié.
UNANSWERED	Messages qui n'ont pas le drapeau \Answered.
UNDELETED	Messages qui n'ont pas le drapeau \Deleted.
UNDRAFT	Messages qui n'ont pas le drapeau \Draft.
UNFLAGGED	Messages qui n'ont pas le drapeau \Flagged.

UNKEYWORD <flag>	Messages qui n'ont pas le mot clé spécifié.
UNSEEN	Messages qui n'ont pas le drapeau \Seen.

Exemple:
C: A282 SEARCH FLAGGED SINCE 1-Feb-1994 NOT FROM "Smith"
S: * SEARCH 2 84 882
S: A282 OK SEARCH completed
S: A282 OK SEARCH completed

- 6.4.5. Commande FETCH

Arguments: Lot de message
Critère de recherche (un ou plusieurs)

Réponses: Réponses de type non marqué : FETCH

Résultat:
OK - fetch effectué
NO - erreur de fetch : ne peut récupérer ces données
BAD - commande inconnue ou arguments non valides

La commande FETCH récupère, dans la boîte aux lettres, les données associées à un message. Les éléments de données que l'on va chercher peuvent être soit de simples atomes soit des listes mise entre parenthèses.

Les éléments de données actuellement définis qui peuvent être récupérés sont :

ALL	Macro équivalente à : (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)
BODY	Structure non-extensible de BODYSTRUCTURE.
BODY[<section>]<<partiel>>	Le texte d'une section particulière de corps. La spécification de la section est une série de zéro ou plusieurs identifiants de morceau délimité par des points. Un identifiant de morceau est soit un numéro de morceau soit l'un des identifiants suivant : HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, et TEXT. Une spécification avec la section à vide fait référence au message entier, en y incluant l'en-tête. Chaque message a eu au moins un numéro de morceau. Les messages Non-[MIME-IMB], et les messages [MIME-IMB] non composé de plusieurs morceau avec aucun message encapsulé, ont seulement le morceau 1. On assigne, pour les messages multipartie, les numéros de morceaux consécutifs au fur et à mesure que ces morceaux apparaissent dans le message. Si un morceau particulier est

de type message ou de type multipartie, ce morceau DOIT (MUST) être indiqué par un point suivi par le numéro de morceau à l'intérieur de la partie multiple ou il est emboîté (sic attention la traduction est plus qu'approximative : -).

Un morceau de type MESSAGE/RFC822 a aussi des numéros imbriqués de morceaux, se référant aux morceaux du corps de la partie MESSAGE (sic).

Les identifiants de morceau HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, et TEXT peuvent être identifiant unique de morceau ou peuvent être préfixé par un ou plusieurs identifiants numériques donné par l'identifiant faisant référence à au morceau de type MESSAGE/RFC822. L'identifiant de partie MIME DOIT (MUST) être préfixé par un ou plusieurs identifiants numériques de morceau.

Les identifiants de morceau HEADER, HEADER.FIELDS, et HEADER.FIELDS.NOT font référence aux en-têtes [RFC-822] du message ou au message [MIME-IMT]MESSAGE/RFC822 encapsulé. HEADER.FIELDS et HEADER.FIELDS.NOT sont suivi par une liste de nom de champs (comme ce qui est défini dans la [RFC-822]), de noms, et retourne une sous partie de l'en-tête. Le sous-ensemble retourné par HEADER.FIELDS contient seulement ceux dont les champs d'en-tête ont un nom de champs qui correspond à un des noms de la liste ; De la même façon le sous-ensemble retourné par HEADER.FIELDS.NOT contient seulement les champs d'en-tête ayant des noms de champs qui ne correspond pas. La correspondance de champs est insensible à la casse, mais doit sinon correspondre exactement. Dans tous les cas, les lignes de blancs qui font les délimitations de blancs entre l'en-tête et le corps sont toujours incluses.

Les identifiants de morceau MIME font référence à l'en-tête [MIME-IMB] du morceau.

L'identifiant du morceau TEXT fait référence au corps du texte du message, en omettant l'en-tête [RFC-822].

Voici un exemple de message complexe avec quelques-uns uns de ces identifiants de morceaux :

```
HEADER (En-tête [RFC-822] du message)
TEXT MULTIPART/MIXED
1 TEXT/PLAIN
2 APPLICATION/OCTET-STREAM
3 MESSAGE/RFC822
3.HEADER (En-tête [RFC-822] du message)
3.TEXT (Corps RFC-822 de type text du mess.)
3.1 TEXT/PLAIN
3.2 APPLICATION/OCTET-STREAM
4 MULTIPART/MIXED
4.1 IMAGE/GIF
4.1.MIME (En-tête [MIME-IMB] pour IMAGE/GIF)
4.2 MESSAGE/RFC822
4.2.HEADER (En-tête [RFC-822] du message)
```

```
4.2.TEXT (Corps RFC-822 de type text du me.)
4.2.1 TEXT/PLAIN
4.2.2 MULTIPART/ALTERNATIVE
4.2.2.1 TEXT/PLAIN
4.2.2.2 TEXT/RICHTEXT
```

Il est possible de prendre une sous chaîne du texte désigné. Ceci est faisable en ajoutant à la fin de l'identifiant de morceau, un caractère inférieur ("<"), la position octale du premier octet désiré, un point, le nombre maximum d'octets désiré, et un caractère inférieur (">"). Si l'octet de départ est situé au-delà de la fin du texte, une chaîne vide est retournée.

Toutes prises où l'on essaie de lire au-delà de la fin du texte est tronquée de façon appropriée. Une prise qui commence à l'octet 0 est retourné en tant une prise partielle, même si cette coupure s'effectue

Remarque: Cela signifie que BODY[]<0.2048> d'un message de 1500 octets retournera BODY[]<0> avec un littéral d'une taille de 1500, et pas BODY[].

Remarque: Une prise d'une sous chaîne d'un identifiant de morceau HEADER.FIELDS ou HEADER.FIELDS.NOT est calculé après avoir après avoir mis en sous-ensemble l'en-tête (sic : after subsetting the header).

Le drapeau \Seen est implicitement mis. Si cela implique que l'on doive changer les marqueurs, ils DOIVENT (SHOULD) être inclue en tant que partie aux réponses à fetch.

BODY.PEEK[<section>]<<partial>>	Une forme alternative à BODY[<section>] qui ne met pas implicitement le drapeau à \Seen.
BODYSTRUCTURE	La structure de corps [MIME-IMB] d'un message. Ceci est fourni par le serveur qui analyse les en-têtes [RFC-822] ainsi que les en-têtes [MIME-IMB].
ENVELOPE	La structure d'enveloppe du message. Ceci est fourni par le serveur qui analyse l'en-tête [RFC-822] en morceau de composant, qui met par défaut divers champs si nécessaire.
FAST	Macro équivalente à: (FLAGS INTERNALDATE RFC822.SIZE)
FLAGS	Les drapeaux qui sont mis sur ce message.
FULL	Macro équivalente à : (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY)

INTERNALDATE	La date interne du message.
RFC822	Fonctionnellement équivalente à BODY[], mais qui diffère de part la syntaxe des données de type non marquées obtenues à la réponse de FETCH. (RFC822 est retourné).
RFC822.HEADER	Fonctionnellement équivalente à BODY.PEEK[HEADER], mais qui diffère de part la syntaxe des données de type non marquées obtenues à la réponse de FETCH. (RFC822.HEADER est retourné).
RFC822.SIZE	La taille[RFC-822] du message.
RFC822.TEXT	Fonctionnellement équivalante à BODY[TEXT], mais qui diffère de part la syntaxe des données de type non marquées obtenues à la réponse de FETCH. (RFC822.TEXT est retourné).
UID	L'identifiant unique du message.

Exemples: C: A654 FETCH 2:4 (FLAGS BODY[HEADER.FIELDS (DATE FROM)])
S: * 2 FETCH ....
S: * 3 FETCH ....
S: * 4 FETCH ....
S: A654 OK FETCH completed

- 6.4.6. Commande STORE

Arguments: Lot de message
Nom d'élément des données du message
Valeur de l'élément des données du message

Réponses: Réponse de type non marqué : FETCH

Résultat: OK - store effectué
NO - erreur de stockage : ne peut stocker ces donnée
BAD - arguments ou commandes non valides

La commande STORE modifie les données associées au message dans la boîte aux lettres. Normalement, STORE retournera la valeur mise à jour des données avec une réponse FETCH de type non marqué. Un suffixe ".SILENT" dans le nom d'élément de donnée permet de ne pas avoir la réponse non marquée FETCH, et le serveur DEVRA (SHOULD) supposer que le client a déterminé de lui-même la valeur mise à jour ou qu'il n'attache guère d'importance à la valeur mise à jour.

Remarque: Sans se soucier de savoir si le suffixe ". SILENT" a été utilisé ou pas, le serveur devra envoyer une réponse FETCH non marquée si un changement, d'origine externe, de drapeaux de message est observé. Le but est que le statut des drapeaux soit facilement

déterminable (without a race condition).

Les éléments de donnée, actuellement définis, qui peuvent être stocké sont :

FLAGS <liste de drapeau>	remplace par l'argument, les drapeaux du message. La nouvelle valeur de ces drapeaux est retournée comme si un FETCH pour ces drapeaux été fait.
FLAGS.SILENT <flag list>	Equivalant à FLAGS, mais sans le fait de retourner une nouvelle valeur.
+FLAGS <flag list>	Ajoute l'argument aux drapeaux du message. La nouvelle valeur des drapeaux est retournée comme si un FETCH de ces drapeaux avait été fait.
+FLAGS.SILENT <flag list>	Equivalant à +FLAGS, mais sans retourner une nouvelle valeur.
-FLAGS <flag list>	Enlève l'argument des drapeaux du message. Les nouvelles valeurs des drapeaux sont retournées comme si un FETCH de ces drapeaux avait été fait.
-FLAGS.SILENT <flag list>	Equivalant à -FLAGS, mais sans retourner une nouvelle valeur.

Exemple:
C: A003 STORE 2:4 +FLAGS (\Deleted)
S: * 2 FETCH FLAGS (\Deleted \Seen)
S: * 3 FETCH FLAGS (\Deleted)
S: * 4 FETCH FLAGS (\Deleted \Flagged \Seen)
S: A003 OK STORE completed

- 6.4.7. Commande COPY

Arguments: Lot de message
Nom de boîte aux lettres

Réponses: Pas de réponse spécifique pour cette commande

Résultat:
OK - copy effectué
NO - copy error : ne peut copier ces messages ou ne peut le faire sur ce nom.
BAD - commande inconnue ou arguments non valides

La commande COPY copie le(s) message(s) spécifié(s) à la fin de la boite aux lettres de destination. Les drapeaux et la date interne du ou des message(s) DEVRA (SHOULD) être conservé lors de la copie.

Si la boîte aux lettres de destination n'existe pas, un serveur DEVRA (SHOULD) retourner une erreur. On NE DEVRA PAS (SHOULD NOT) créer automatiquement la boîte aux lettres. A moins que l'on soit sûr de ne pas pouvoir créer la boîte aux lettres de destination, le serveur DOIT (MUST) envoyer le code de réponse "[TRYCREATE]" en tant que préfixe du texte de la réponse NO qui est

de type non marqué. Cela indique au client qu'il peut tenter une commande CREATE et réessayer la COPY si le CREATE est un succès.

Si la commande COPY est un échec pour quelques raisons que se soient, les implémentations serveur DOIVENT (MUST) remettre en l'état la boîte aux lettres de destination, dans l'état où elle était avant la tentative de COPY.

Exemple: C: A003 COPY 2:4 MEETING
S: A003 OK COPY completed

- 6.4.8. Commande UID

Arguments: Nom de commande
Arguments de commande

Réponses: Réponses non marquées : FETCH, SEARCH

Résultat: OK - commande UID effectuée
NO - erreur dans la commande UID
BAD - commande inconnue ou arguments non valides

La commande UID a 2 formes. Pour la première forme, elle prend pour arguments, une commande COPY, FETCH, ou STORE avec les arguments approprié à la commande associée. Cependant, les nombres pour les arguments de lot de message sont constitués par les identifiants uniques au lieu des numéros de séquence de message.

Dans la deuxième forme, la commande UID prends la commande SEARCH avec les arguments de la commande SEARCH. L'interprétation de ces arguments est la même qu'avec SEARCH; Cependant, les nombres retournés dans la réponse SEARCH pour une commande UID SEARCH sont les identifiants uniques au lieu des numéros de séquence de message. Par exemple, la commande UID SEARCH 1:100 UID 443:557 retournes les identifiants uniques correspondant à l'intersection des lots de numéro de séquence de message établi à 1:100 et les lots avec l'UID à 443:557.

Les ensembles (range) de lots de message sont permis : Cependant, il n'y a aucune garantie a ce que les identifiants uniques soient contiguë. Un identifiant unique n'existant pas à l'intérieur d'un ensemble de lot de message est ignoré sans qu'un message d'erreur soit généré.

Le nombre après le "*" dans la réponse FETCH de type non marqué est toujours un numéro de séquence de message, et pas d'identifiant unique, même pour une réponse à une commande UID. Cependant, les implémentations de serveurs DOIVENT (MUST) inclure implicitement l'élément de donnée de message UID, comme faisant partie d'une quelconque réponse FETCH généré par une commande UID, sans regarder si l'UID était spécifié en tant qu'élément de donnée de message à un FETCH.

Exemple: C: A999 UID FETCH 4827313:4828442 FLAGS
S: * 23 FETCH (FLAGS (\Seen) UID 4827313)
S: * 24 FETCH (FLAGS (\Seen) UID 4827943)
S: * 25 FETCH (FLAGS (\Seen) UID 4828442)
S: A999 UID FETCH completed

- 6.5. Commandes Clients - Expérimentale/Annexe

- 6.5.1. Commande X<atom>

  Arguments: Défini par l'implémentation

  Réponses: Défini par l'implémentation

  Résultat:
  OK - commande effectuée
  NO - échec
  BAD - commande inconnue ou arguments non valides

  Toutes commandes préfixées avec un X est une commande expérimentale. Les commandes qui ne font pas partie de cette spécification, une révision à cette spécification de niveau standard ou de niveau standards-track, ou un protocole expérimental approuvé par l'IESG, doivent utiliser le préfixe X.

  Toutes réponses ajoutées de type non marqué effectué par une commande expérimentale DOIT (MUST) aussi être préfixé par un X. Les implémentations de serveurs NE DOIVENT PAS (MUST NOT) envoyer de telles réponses non marquées, a moins que le client l'ait demandé en exécutant la commande expérimentale associée.

  Exemple:
  C: a441 CAPABILITY
  S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4 XPIG-LATIN
  S: a441 OK CAPABILITY completed
  C: A442 XPIG-LATIN
  S: XPIG-LATIN ow-nay eaking-spay ig-pay atin-lay
  S: A442 OK XPIG-LATIN ompleted-cay

- **7. Réponses Serveurs**

  Les réponses serveur sont de trois ordres possibles : les réponses de statut, les données serveurs, et les demandes de continuation de commande. Les informations contenues dans une repense de serveur et identifié par "contenu :" dans les descriptions de réponses ci-dessous, sont décrites par fonction et non par syntaxe. La syntaxe précise des réponses serveurs est décrite dans la section Syntaxe Formelle.

  Le client DOIT (MUST) être prêt à accepter à tout moment n'importe quelle réponse.

  Les réponses de statut peuvent être de type marqué ou non marqué. Les réponses de statut marqué indiquent le résultat de l'achèvement (statut OK, NO, ou BAD) d'une commande client, et ont un marqueur correspondant à la commande.

  Certaines réponses de statut, et toutes les données serveurs, sont non marquées. Une réponse non marquée est indiquée par le signe "*" à la place d'un marqueur. Les réponses de statut de type non marqué signale la présentation serveur, ou les statuts serveur qui n'indiquent pas la terminaison d'une commande (par exemple, une alerte signalant l'arrêt imminent du système). Pour des raisons historiques, les réponses de données serveur de type non marqué sont aussi appelées "données non sollicitées", bien que strictement parlant, seule les données serveurs unilatérales sont vraiment "non sollicitée".

Les données serveur avéré DOIVENT (MUST) être enregistrée par le client quand elles sont reçues ; Ceci est noté dans la description de ces données. De telle donnée transportent des informations critiques qui affectent l'interprétation de toutes les commandes et réponses qui suivront (par exemple les mises à jours qui renvoient à la création ou à la destruction des messages).

D'autres données serveur DEVRAIT (SHOULD) être enregistré pour constituer des renseignements ultérieurs ; Si le client n'a pas besoin d'enregistrer les données, ou si l'enregistrement des données ne portent pas sur un sujet clair (par exemple une réponse SEARCH quand aucune commande SEARCH est en cours), on devra ignorer les données.

Un exemple de données serveur unilatérales de type non marqué apparaît quand la connexion IMAP est en état sélectionné. En état sélectionné, le serveur contrôle la boîte aux lettres à la recherche de nouveaux messages en tant partie de la commande en cours d'exécution. Normalement, cela fait partie de l'exécution de toutes commandes ; de ce fait, une commande Noop suffit pour contrôler l'arrivée de nouveaux messages. Si de nouveaux messages sont trouvés, le serveur envoie des réponses EXISTS et RECENT de type non marqué renvoyant la nouvelle taille de la boîte aux lettres. Les implémentations serveurs qui offrent des accès simultanés et multiples à la même boîte aux lettres DEVRAIT (SHOULD)aussi envoyer des réponses EXPUNGE et FETCH unilatérale de type non marquée, si un autre agent change l'état de n'importe quel drapeau de message ou expunge n'importe quels messages.

Les réponses de demande de commande de continuation utilisent le signe "+" à la place d'un marqueur. Ces réponses sont envoyées par le serveur pour indiquer l'acceptation une commande client incomplète et son "empressement" à recevoir le reste de la commande.

- 7.1. Réponses serveur - Réponses de statut

Les réponses de statut sont OK, NO, BAD, PREAUTH et BYE. OK, NO, et BAD peuvent être de type marqué ou non marqué. PREAUTH et BYE sont toujours non marquées.

Les statuts de réponses PEUVENT (MAY) inclure un "code réponse" OPTIONNEL (OPTIONNAL). Un code réponse comprends des données entre crochets droits ([] "square bracket") sous forme d'atome, avec la possibilité quelles soient suivies par un espace et des arguments. Les codes réponses contiennent des informations supplémentaires ou des codes de statut pour le logiciel client situé derrière l'état OK/NO/BAD, et sont définis quand il y a une action spécifique qu'un client puisse faire en fonction des informations supplémentaires.

Les codes réponses actuellement définis sont :

ALERT	Le texte lisible par un humain contient une alerte qui DOIT (MUST) être présenté à l'utilisateur de façon à ce que capte l'attention de l'utilisateur.
NEWNAME	Suivi par un nom de boîte aux lettres et par un nouveau de boîte aux lettres. Un EXAMINE ou un SELECT a échoué parce que le nom de la boîte aux lettres cible n'existe plus du fait que cette dernière a été renommée sous le nouveau nom de boîte aux lettres. C'est une indication donnée au client pour lui signifier que l'opération peut être un succès si le SELECT ou l'EXAMINE est rééditée avec le nouveau nom de boîte aux lettres.
PARSE	Le texte lisible par un être humain signale une erreur pour ce qui est de l'analyse syntaxique de l'en-tête [RFC-822] ou pour les en-têtes [MIME-IMB] d'un message dans la boîte aux lettres.

PERMANENTFLAGS	Suivi par une liste mise entre parenthèse de drapeaux, montrent parmi les drapeaux connus, quels sont les drapeaux que le client peut changer de façon permanente. N'importe quels des drapeaux qui sont dans la réponse non marquée FLAGS, exceptés ceux de la liste PERMANENTFLAGS, ne peuvent pas être fixés de façon permanente. Si le client a l'intention de faire un STORE d'un drapeau qui n'est pas dans la liste des PERMANENTFLAG, le serveur le rejettera soit avec une réponse NO, ou enregistrera l'état uniquement pour le restant de la session en cours. La liste PERMANENTFLAGS peut aussi inclure le drapeau spécial *, qui indique qu'il est possible de créer de nouveaux mot-clés en ayant pour intention de stocker ces drapeaux dans la boîte aux lettres.
READ-ONLY	La boîte aux lettres est sélectionnée en lecture seule, ou ses accès, durant la sélection ont changé, de lecture-écriture en lecture seulement.
READ-WRITE	La boîte aux lettres est sélectionnée en lecture-ecriture, ou ses accès, durant la sélection, ont changé, de lecture seule en lecture écriture.
TRYCREATE	Une tentative d'APPEND ou de COPY a échoué du fait que la boîte aux lettres cible n'existe pas (contrairement à certaines autres raisons). C'est l'indication pour le client que l'opération peut être un succès si la boîte aux lettres est en premier crée par la commande CREATE.
UIDVALIDITY	Suivi par un nombre décimal, indique la valeur de validité d'identifiant unique./td>
UNSEEN	Suivi par un nombre décimal, indique le numéro du premier message qui n'a pas le drapeau à \seen.

Des codes de réponses supplémentaires définies pour des implémentations particulières de serveur ou de client DOIVENT (SHOULD) être préfixé par un "X" jusqu'à qu'ils soient ajoutés à une révision de ce protocole. Les implémentations client DEVRAIT (SHOULD) ignorer les codes réponses qu'ils ne reconnaissent pas.

- 7.1.1. Réponse OK

Contenu: Code réponse OPTIONNEL (OPTIONAL)
Texte lisible par un humain.

La réponse OK indique un message d'information provenant du serveur. Quand, il est marqué, il indique la bonne fin de la commande associée. Le texte lisible par un humain PEUT (MAY) être présenté à l'utilisateur en tant que message d'information. La forme non marquée indique un message seulement d'information. La nature de l'information PEUT (MAY) être indiqué par un code de réponse.

La forme non marquée est aussi utilisée comme une des trois présentations de bienvenue possible au commencement de la connexion. Cela indique que la connexion n'est pas encore authentifiée et qu'une commande de LOGIN est indispensable.

Exemple: S: * OK IMAP4rev1 server ready

C: * OK IMAP4rev1 server ready
S: * OK [ALERT] System shutdown in 10 minutes
S: A001 OK LOGIN Completed

- 7.1.2. Réponse NO

Contenu: Code réponse OPTIONNEL (OPTIONAL)
Texte lisible par un humain.

La réponse NO indique un message d'erreur de fonctionnement du serveur. Quand, il est marqué, elle indique que la mauvaise fin de la commande qui lui a été associé. La forme non marquée indique un avertissement ; la commande peut encore se finir avec succès. Le texte compréhensible par un utilisateur décrit la situation.

Exemple: C: A222 COPY 1:2 owatagusiam
S: NO Disk is 98% full, please delete unnecessary data
S: A222 OK COPY completed
C: A223 COPY 3:200 blurdybloop
S: * NO Disk is 98% full, please delete unnecessary data
S: * NO Disk is 99% full, please delete unnecessary data
S: A223 NO COPY failed: le disque est plein

- 7.1.3. Réponse BAD

Contenu: Code réponse OPTIONNEL (OPTIONAL)
Texte visible par un humain.

La réponse BAD indique un message d'erreur du serveur. Quand elle est marquée, cela rend compte d'une erreur au niveau du protocole dans la commande du client. La forme qui est de type marqué, indique la commande qui a provoqué l'erreur. La forme de type non marquée indique une erreur de protocole pour laquelle la commande associé ne peut être déterminée. Cela peut aussi indiquer une défaillance interne du serveur. Le texte lisible pour un humain en décrit les circonstances.

Exemple: C: ...very long command line...
S: * BAD Command line too long
C: ...empty line...
S: * BAD Empty command line
C: A443 EXPUNGE
S: * BAD Disk crash, attempting salvage to a new disk!
S: * OK Salvage successful, no data lost
S: A443 OK Expunge completed

- 7.1.4. Réponse PREAUTH

Contenu: Code réponse OPTIONNEL (OPTIONAL)
Texte compréhensible par un humain

La réponse PREAUTH est toujours non marquée, et c'est une des trois présentations possibles de

bienvenue au démarrage de la connexion. Elle indique que la connexion a déjà été authentifiée par des moyens externes et, ainsi, aucune commande de LOGIN n'est nécessaire.

Exemple: S: * PREAUTH IMAP4rev1 server logged in as Smith

- 7.1.5. Réponse BYE

Contenu: Code réponse OPTIONNEL (OPTIONAL)
Texte lisible par un humain

La réponse BYE est toujours non marquée, et indique que le serveur est sur le point de fermer la connexion. Le texte lisible par l'homme PEUT (MAY) être afficher à l'utilisateur dans un rapport de statut pour le client. La réponse BYE est envoyée pour une des ces quatre conditions :

1) En tant que faisant partie d'une séquence de sortie normale (logout). Le serveur fermera la connexion après avoir envoyer la réponse OK de type marqué lors de la commande de LOGOUT.

2) Comme l'annonce d'un arrêt en catastrophe. Le serveur ferme immédiatement les connexions.

3) Comme la déclaration d'un autologout d'inactivité. Le serveur ferme toutes les connexions immédiatement.

4) En tant qu'une des trois possibles présentations de bienvenue lors du démarrage de bienvenue, ce qui indique que le serveur n'est pas disposer à accepter une connexion pour ce client. Le serveur ferme la connexion immédiatement.

La différence entre un BYE qui se passe en tant que partie d'une séquence normale de LOGOUT (le premier cas) et le BYE qui s'effectue du fait d'un échec (les trois autres cas) est que les connexions se ferment immédiatement dans le cas de l'échec.

Exemple: S: * BYE Autologout; inoccupé depuis trop longtemps

- 7.2. Réponses Serveur - Statut de serveur et de boîtes aux lettres

Ces réponses sont toujours de type non marqué. Voici comment les données de statut serveur et de boîte aux lettres sont transmises du serveur vers le client. Beaucoup de ces réponses ont pour caractéristique d'être le résultat d'une commande portant le même nom.

- 7.2.1. Réponse CAPABILITY

Contenu: Liste de fonctionnalité

La réponse CAPABILITY survient en tant que résultat de la commande CAPABILITY. La liste des capacités contient une liste de noms, séparés par un espace, des fonctionnalités que le serveur supporte. La liste des capacités DOIT (MUST) inclure l'entité "IMAP4rev1".

Une désignation de fonctionnalité qui commence par "AUTH=" indique que le serveur supporte ce mécanisme d'authentification particulier.

D'autres désignations de fonctionnalités montre que le serveur supporte une extension, une révision

ou un amendement au protocole IMAP4rev1. Les réponses serveurs DOIVENT (MUST) se conformer à ce document, jusqu'au moment où le client émet une commande qui utilise la capacité associée.

Les noms de fonctionnalités DOIVENT (MUST) soient commencer par un "X" ou soient sont standard (à la norme), soient sont conforment aux extensions standards-track IMAP4rev1, soient sont des révisions, ou des corrections enregistrées auprès de IANA. Un serveur NE DOIT PAS (MUST NOT)offrir des noms de capacités non-standard ou non enregistrée, à moins que de tels noms soit préfixé par un "X".

Les implémentations clients, NE DEVRAIENT PAS (SHOULD NOT) avoir besoin comme nom de fonctionnalité d'autre chose que "IMAP4rev1", et DOIVENT (MUST) ignorer tous les noms inconnus de capacité.

Exemple: S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4 XPIG-LATIN

- 7.2.2. Réponse LIST

Contenu: Attribut au nom
Délimiteur hiérarchique
Nom

La réponse LIST arrive en tant que résultat à une commande LIST. Elle retourne un simple nom qui correspond à la spécification de LIST.

Quatre attributs au nom sont définis :

\Noinferiors Il est impossible à des niveaux fils dans la hiérarchie d'exister au-dessous de ce nom. Aucun niveau enfant n'existe et dans le futur, aucun ne pourra être créer.

\Noselect Il est impossible d'utiliser ce nom en tant que boîte aux lettres sélectionnable.

\Marked La boite aux lettres a été marquée "interesting" intéressante ("interesting") par le serveur ; La boite aux lettres contiens probablement des messages qui ont été ajouté depuis qu'elle a été sélectionnée la dernière fois.

\Unmarked La boîte aux lettres ne contient pas de messages qui ont été ajouté depuis qu'elle a été sélectionnée la dernière fois.

Si, il n'est pas réalisable pour un serveur de déterminer si la boîte aux lettres est intéressante ("interesting") ou ne l'est pas, ou si le nom est un nom de type \Noselect, le serveur ne DOIT pas (SHOULD NOT) envoyer de \Marked ou \Unmarked.

Le délimiteur hiérarchique est un caractère utilisé pour délimiter les niveaux de hiérarchie dans un nom de boîte aux lettres. Un client peut l'utiliser pour créer des boîtes aux lettres enfants, et pour chercher dans la hiérarchie de nommage des niveaux plus hauts ou plus bas. Tous les enfants d'un nœud de niveau supérieur DOIVENT (MUST) utiliser le même caractère séparateur. Un délimitateur de hiérarchie NIL signifie qu'il n'y a pas de hiérarchie ; Le nom est un nom à "plat".

Le nom représente une hiérarchie de gauche à droite non ambiguë, et DOIT (MUST) être valide pour qu'il puisse être utilisé comme une référence pour les commandes LIST et LSUB. A moins que \Noselect soit signalé, le nom DOIT aussi être valide comme argument pour les commandes, telle

que SELECT, qui accepte (comme argument) les noms de boîte aux lettres

Exemple: S: * LIST (\Noselect) "/" ~/Mail/foo

- 7.2.3. Réponse LSUB

Contenu: Attribut au nom
Délimiteur hiérarchique
Nom

La réponse LSUB apparaît comme résultat à une commande LSUB. On retourne un seul nom qui corresponds à la spécification de LSUB. Il peut y avoir plusieurs réponses LSUB pour une seule commande. Les informations sont du même format que celles à une réponse LIST.

Exemple: S: * LSUB () "." #news.comp.mail.misc

- 7.2.4. Réponse STATUS

Contenu: Nom
Liste de statut mis entre parenthèse.

La réponse STATUS apparaît en tant que résultat de la commande STATUS. On retourne le nom de la boîte aux lettres qui a la description de statut qui correspond au statut de la boîte aux lettres demandées.

Exemple: S: * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)

- 7.2.5. Réponse SEARCH

Contenu: Zéro ou plusieurs numéros

La réponse SEARCH apparaît comme résultat à une commande SEARCH ou UID SEARCH. Le(s) numéro(s) font référence à ceux des messages qui correspondent au critère de recherche. Pour SEARCH, ce sont des numéros de séquences de message. Pour UID SEARCH, ce sont des identifiants uniques. Chaque nombre est délimité par un espace.

Exemple: S: * SEARCH 2 3 6

- 7.2.6. Réponse FLAGS

Contenu: Liste de drapeau mise entre parenthèse

La réponse FLAGS apparaît comme résultat à une commande SELECT ou EXAMINE. La liste de drapeaux mis entre parenthèse identifie les drapeaux (aux moins, les drapeaux définis par le système) qui sont applicables pour cette boîte aux lettres. Des drapeaux autres que les drapeaux systèmes peuvent aussi exister, et sont dépendant de l'implémentation du serveur.

La mise à jour à partir de la réponse FLAGS DOIT (MUST) être enregistrée par le client.

Exemple: S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)

- 7.3. Réponses Serveur - Taille de la boîte aux lettres

  Ces réponses sont toujours de type non marqué. On indique par cela comment les changements de taille de boîtes aux lettres sont transmis du serveur vers le client. Un nombre suit immédiatement le drapeau "*" et ce nombre représente un compteur de message.

- 7.3.1. Réponse EXISTS

  Contenu: Aucun

  La réponse EXISTS donne le nombre de message qui sont dans la boîte aux lettres. Cette réponse apparaît comme étant le résultat d'une commande SELECT ou EXAMINE, et si la taille de la boîte aux lettres a changé (par exemple par de nouveaux courriers).

  La mise à jour de la réponse EXIST DOIT (MUST) être enregistrée par le client.

  Exemple: S: * 23 EXISTS

- 7.3.2. Réponse RECENT

  Contenu: Aucun

  La réponse RECENT donne le nombre de message avec le drapeau mis à \Recent. Cette réponse est le résultat d'une commande SELECT ou EXAMINE, et seulement si la taille de la boite à lettre a changé (par exemple de nouveau message).

  Remarque: Il n'est pas garanti que les numéros de séquences de message les plus récents constitue une étendue contiguë des n messages les plus hauts dans la boîte aux lettres (ou n est la valeur rapportée par la réponse RECENT). Voici des exemples de situations pour lesquels ce n'est pas le cas : Plusieurs clients qui ont la même boîte aux lettres ouverte (la première session a être notifiée le verra comme récent, les autres le verront probablement comme non-recent), ou quand la boîte aux lettres est réordonnée par un agent non-IMAP.

  Le seul moyen fiable d'identifier des messages récents est de chercher les drapeaux de message pour quelles le drapeau a été mis à \recent, ou de faire un SEARCH RECENT.

  La mise à jour de la réponse RECENT DOIT (MUST) être enregistrée par le client.

  Exemple: S: * 5 RECENT

- 7.4. Réponses Serveur - Message de statut

  Ces réponses sont toujours de type non marqué. C'est de cette manière que les données de message sont transmises du serveur vers le client, souvent en tant que résultat d'une commande pourtant le même nom. Immédiatement après le signe "*" il y a un nombre qui représente un numéro de séquence de message.

- 7.4.1. Réponse EXPUNGE

  Contenu: Aucun

  La réponse EXPUNGE donne le numéro de séquence de message qui a été enlevé de façon

définitive de la boîte aux lettres. Les numéros de séquence de message pour tout les messages suivant sont décrementé de un, et cette décrementation se répercute sur les numéros de séquence de message pour les réponses suivantes (y compris les autres réponses EXPUNGE non marquées).

En tant le résultat direct de la règle de décrementation, les numéros de séquence de message qui apparaissent dans les réponses successives d'expunge dépendent de la manière dont les messages sont supprimés, c'est à dire si ce sont des numéros les plus petits du fait vers les numéros les plus grand, ou des numéros les plus grands vers les numéros les plus petits. Par exemple, si les 5 derniers messages d'une boîte aux lettres de 9 messages sont expunges ; un serveur du plus petit au plus bas enverra cinq réponses EXPUNGE non marquée pour le message de séquence numéro 5, tandis que le serveur du plus grand au plus petit enverra successivement les réponses EXPUNGE non marquée pour les numéros de séquence de message 9,8,7,6, et 5.

Une réponse EXPUNGE NE DOIT PAS (MUST NOT) être envoyée quand aucune commande est en cours ; Ni même pendant une réponse à une commande FETCH, STORE ou SEARCH. Cette règle est nécessaire pour empêcher la perte de synchronisation des numéros de séquence entre le client et le serveur.

La mise à jour que l'on récupère par EXPUNGE DOIT (MUST) être enregistrée par le client.

Exemple: S: * 44 EXPUNGE

- 7.4.2. Réponse FETCH

Contenu: Donnée de message

La réponse FETCH retourne, au client, des données faisant partie d'un message. Les données vont de pair avec les noms d 'élément de données et leurs valeurs mise entre parenthèse. Cette réponse arrive comme étant le résultat d'une commande FETCH ou STORE, mais aussi en tant que décision unilatérale du serveur (par exemple lors de mises à jours de drapeau).

Les actuels éléments de donnée sont :

BODY	une forme de BODYSTRUCTURE sans les données d'extension.
BODY[<section>]<<origin_octet>>	Une chaîne précise le contenu du corps de la section spécifiée. La chaîne DEVRA (SHOULD) être interprétée par le client en respectant, la manière dont le contenu a été codé pour l'encodage de transfert, le type du corps, et le sous-type. Si l'octet d'origine est précisé, cette chaîne est une sous chaîne du contenu de corps, qui commencera à cet octet d'origine. Cela signifie que BODY[]<0> PEUT (MAY) être tronqué, mais que BODY[] n'est JAMAIS (NEVER) tronqué. Les données textuelles en 8-bit sont autorisés si l'identifiant [CHARSET] fait partie de la liste des paramètres mis entre parenthèse pour cette section. Notez que les en-têtes (les identifiants de partie HEADER ou MIME, ou la portion d'en-tête de la partie MESSAGE/RFC822), DOIVENT (MUST) être sur 7-bit ; Les caractères en 8-bit ne sont pas autorisé dans les en-têtes. Notez aussi que les lignes blanches

(blank line) à la fin de l'en-tête est toujours incluse dans les données de l'en-tête.

Les données non textuelles, tel que les données de type binaire, DOIVENT (MUST) être transféré codé dans une forme textuelle tel que celle de BASE64 avant d'être envoyés au client. Pour retrouver les données binaires originelles, le client DOIT (MUST) décoder la chaîne qui a été transférée codé.

BODYSTRUCTURE

Une liste mise entre parenthèses qui décrit la structure de corps [MIME-IMB] d'un message. C'est "fourni" par le serveur en faisant une analyse des champs d'en-tête [MIME-IMB], en mettant divers champs par défaut, si nécessaire. Par exemple, un simple message texte de 48 lignes et de 2279 octets peut avoir une structure de corps du type : ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48)

On indique les parties de type multiples par des parenthèses imbriquées. Au lieu d'avoir un type de corps en tant que premier élément d'une liste mise entre parenthèse, on peut avoir un corps imbriqué. Le second élément d'une liste entre parenthèse est le sous type de multipartie (mixed, digest, parallel, alternative, etc.).

Par exemple, un message en deux parties, comprenant un texte et un attachement de texte encodé-BASE64 peut avoir une structure de corps de la forme : (("TEXT" "PLAIN" ("CHARSET""US-ASCII") NIL NIL "7BIT" 1152 23)("TEXT" "PLAIN" ("CHARSET" "US-ASCII" "NAME" "cc.diff") "<960723163407.20117h@cac.washington. edu>" "Compiler diff" "BASE64" 4554 73) "MIXED"))

Les données d'extensions suivent le sous type de partie multiple. L'extension de donnée n'est jamais retournée par le fech BODY, mais peut être retournée par un fetch BODYSTRUCTURE. Les données d'extensions, si elles sont présentent, DOIVENT (MUST) être dans l'ordre défini.

Les extensions des données d'un corps à partie multiple sont dans l'ordre suivant :

liste de paramètre de corps mis entre parenthèse

Une liste entre parenthèse de binômes de valeurs/attribut [par exemple ("foo" "bar" "baz" "rag") ou "bar" est la valeur de "foo" et "rag" est la valeur de "baz"] comme ce qui est défini dans [MIME-IMB].

body disposition

Une liste mise entre parenthèse, comprenant une chaîne de type disposition suivi par une liste mise entre parenthèse de binôme de distribution

attribut/valeur. Le type de disposition et les noms d'attribut seront définis dans une future révision standards-track de [DISPOSITION].

body language

Une chaîne ou une liste entre parenthèse donnant la langue du corps comme ce qui est défini dans [LANGUAGE-TAGS].

Toutes les données d'extension suivantes ne sont pas encore définies dans cette version du protocole. De telles données d'extension peuvent consister en zéro ou plusieurs NIL, chaînes, nombres, ou de potentielles listes entre parenthèses imbriquées comprenant de telle données. Les implémentations client qui font un fetch BODYSTRUCTURE DOIVENT (MUST) être préparer à accepter ces données étendues. Les implémentations serveurs NE DOIVENT (MUST NOT) PAS envoyer des données faisant partie des extensions avant qu'elles ne soient définies par une révision de ce protocole.

Les champs de base des régions de corps de type non multitpartie sont dans l'ordre qui suit :

body type

Une chaîne qui donne le nom du type du support du contenu comme c'est défini dans [MIME-IMB].

body subtype

Une chaîne qui donne le nom de sous type du contenu défini dans [MIME-IMB].

Liste entre parenthèse de paramètre de corps

Une liste mise entre parenthèse de binôme attribut/valeur [par exemple ("foo" "bar" "baz" "rag") ou "bar" est la valeur de "foo" et "rag" est la valeur de "baz"] comme ce qui est défini dans [MIME-IMB].

body id

Une chaîne donnant l'id du contenu comme c'est défini dans [MIME-IMB].

body description

Une chaîne donnant la description du contenu comme c'est défini dans [MIME-IMB].

body encoding

Une chaîne donnant l'encodage de transfert du contenu qui a été transféré comme c'est défini dans [MIME-IMB].

body size

Un nombre donnant la taille du corps en octet. Notez que cette taille est la taille dans son codage de transfert et non la taille après décodage.

Un type de corps de type MESSAGE et de sous type RFC822 contient, immédiatement après les champs de base, la structure d'enveloppe, la structure du corps, et la taille en ligne de texte du message encapsulé.

Un type de corps de type TEXT contient, immédiatement après les champs de base, la taille du corps en ligne de textes. Notez que cette taille est la taille de son contenu transféré encodé et non la taille après tout décodage.

Les données d'extensions suivent les champs de base et les champs de type spécifique listé au-dessous. Les données d'extensions ne sont jamais retournées par le fetch BODY, mais peuvent être retourné par un fetch BODYSTRUCTURE. Les extensions de données, si elles sont présente, DOIVENT (MUST) être dans l'ordre défini.

Les données d'extensions d'une partie d'un corps de type non multipartie sont dans l'ordre suivant :

body MD5

Une chaîne donnant la valeur MD5 du corps comme défini dans [MD5].

body disposition

Une liste mis entre parenthèse avec le même contenu et la même fonction que la disposition du corps pour une partie de corps de type multipartie.

body language

Une chaîne ou une liste mise entre des parenthèses donnant la langue du corps de message comme ce qui est défini dans [LANGUAGE-TAGS].

Toutes les données d'extensions ne sont pas encore définies dans cette version du protocole, et risque sûrement d'être comme décrites au-dessus en tant que donnée d'extension de type multipartite.

ENVELOPE

Une liste entre parenthèse qui décrit la structure d'un message. C'est fourni par le serveur en analysant les en-têtes [RFC-822] des morceaux qui les composent, mettant par défaut de

	champs varié si nécessaire. Les champs de la structure de l'enveloppe sont dans l'ordre suivant : date, subject, from, sender, reply-to, to, cc, bcc, in-reply-to, et message-id. Les champs date, subject, in-reply-to, et message-id sont des chaînes. Les champs from, sender, reply-to, to, cc, et bcc sont des listes de structure d'adresses mises entre parenthèses. Une structure d'adresse est une liste mise entre parenthèse qui décrit une adresse de courrier électronique. Les champs d'une adresse de courrier électronique sont dans l'ordre suivant : nom personne, "at" (@) liste de domaine [SMTP](route source), nom de boîte aux lettres, et nom d'hôte. La syntaxe de groupe [RFC-822] est indiqué par une syntaxe de groupe qui est indiqué par une forme spéciale de structure d'adresse pour lequel le champ de nom d'hôte est NIL. Si le champ du nom de boîte aux lettres est aussi NIL, c'est la fin d'un marqueur de groupe (point virgule en syntaxe RFC 822). Si le champ du nom de boîte aux lettres est non-NIL, c'est le commencement d'un marqueur de groupe, et le champ de boite à lettre contiens la locution du nom de groupe. Tout champ d'une structure d'adresse ou d'enveloppe qui n'est pas applicable est présente en tant qu'un NIL. Notez que le serveur DOIT (MUST) mettre par défaut les champs de réponse à (reply-to)et d'expédition (sender) à partir du champ from (de) ; Un client n'est censé savoir le faire.
FLAGS	Une liste mise entre parenthèse de drapeaux qui ont été établi pour ce message.
INTERNALDATE	Une chaîne représentant la date interne d'un message
RFC822	Equivalent à BODY[].
RFC822.HEADER	Equivalent à BODY.PEEK[HEADER].
RFC822.SIZE	Un nombre exprimant la taille [RFC-822] du message.
RFC822.TEXT	Equivalent à BODY[TEXT].
UID	Un nombre formulant l'identifiant unique d'un message./td>

Exemple: S: * 23 FETCH (FLAGS (\Seen) RFC822.SIZE 44827)

- 7.5. Réponses Serveur - Demande de continuation de commande

La réponse de demande de continuation de commande est indiquée par le signe "+" à la place d'un marqueur. Cette forme de réponse indique que le serveur est près à accepter la continuation d'une

commande de la part du client. Le reste de cette réponse est une ligne de texte.

Cette réponse est utilisée dans la commande AUTHORIZATION pour transmettre les données serveur à un client, et les demandes de données supplémentaires du client. Ces réponses sont aussi utilisées si un argument à n'importe quelle commande est utilisée comme un littéral

Le client n'a pas l'autorisation d'envoyer les octets d'un littéral à moins que le serveur n'indique qu'il les attend. Cela permet au serveur d'effectuer les commandes et de rejeter les erreurs ligne à ligne. Le reste de la commande, y compris le CRLF qui termine une commande suit les octets du littéral. S'il y a des arguments de la commande qui soient additionnel, l'octet littéral est suivi par un espace et par ces arguments.

Exemple:
C: A001 LOGIN {11}
S: + Ready for additional command text
C: FRED FOOBAR {7}
S: + Ready for additional command text
C: fat man
S: A001 OK LOGIN completed
C: A044 BLURDYBLOOP {102856}
S: A044 BAD No such command as "BLURDYBLOOP"

- **8. Exemple de connexion IMAP4rev1**

Ce qui suit est la transcription d'une connexion IMAPrev1. Une longue ligne dans cet exemple est coupée pour des raisons de clarté éditoriale.

Exemple:
S: * OK IMAP4rev1 Service Ready
C: a001 login mrc secret
S: a001 OK LOGIN completed
C: a002 select inbox
S: * 18 EXISTS
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * 2 RECENT
S: * OK [UNSEEN 17] Message 17 is the first unseen message
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: a002 OK [READ-WRITE] SELECT completed
C: a003 fetch 12 full
S: * 12 FETCH (FLAGS (\Seen) INTERNALDATE "17-Jul-1996
RFC822.SIZE 4286 ENVELOPE ("Wed, 17 Jul 1996 02:23:25 -0700 (PDT)"
"IMAP4rev1 WG mtg summary and minutes"
(("Terry Gray" NIL "gray" "cac.washington.edu"))
("Terry Gray" NIL "gray" "cac.washington.edu"))
(("Terry Gray" NIL "gray" "cac.washington.edu"))
((NIL NIL "imap" "cac.washington.edu"))
((NIL NIL "minutes" "CNRI.Reston.VA.US")
("John Klensin" NIL "KLENSIN" "INFOODS.MIT.EDU")) NIL NIL
"<B27397-0100000@cac.washington.edu>")

BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 3028
92))

S: a003 OK FETCH completed

C: a004 fetch 12 body[header]

S: * 12 FETCH (BODY[HEADER] {350}

S: Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)

S: From: Terry Gray <gray@cac.washington.edu>

S: Subject: IMAP4rev1 WG mtg summary and minutes

S: To: imap@cac.washington.edu

S: cc: minutes@CNRI.Reston.VA.US, John Klensin
<KLENSIN@INFOODS.MIT.EDU>

S: Message-Id: <B27397-0100000@cac.washington.edu>

S: MIME-Version: 1.0

S: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII

S:

S:)

S: a004 OK FETCH completed

C: a005 store 12 +flags \deleted

S: * 12 FETCH (FLAGS (\Seen \Deleted))

S: a005 OK +FLAGS completed

C: a006 logout

S: * BYE IMAP4rev1 server terminating connection

S: a006 OK LOGOUT completed

- **9. Syntaxe formelle**

La spécification de la syntaxe qui suit utilise la notation augmentée de la Forme Backus-Naur (BNF) comme spécifié dans la [RFC-822] avec une exception : Le délimiteur utilisé avec le constructeur "#" est un simple espace (SPACE) et non une ou plusieurs virgules.

Dans le cas où des règles optionnelles ou alternatives pour lesquels une règle chevaucherait une règle plus récente, la règle qui est listée la première DOIT (MUST) être prioritaire. Par exemple, quand "\Seen" est analysé comme étant un drapeau, il s'agit du nom de drapeau \Seen et non d'une extension de drapeau, bien que "\seen" puisse être analysé comme une extension de drapeau (flag_extension). Quelques cas (mais pas tous) illustrant cette règle sont notés au-dessous.

Excepté ce qui est noté ailleurs, tous les caractères alphabétiques sont insensibles à la casse. L'utilisation de caractère majuscule ou minuscule pour définir les chaînes de marques (token string) est uniquement faite pour des raisons de clarifications éditoriales. Les implémentations DOIVENT (MUST) accepter ces chaînes de manière à être insensible à la casse.

```
address         ::= "(" addr_name SPACE addr_adl SPACE
                    addr_mailbox SPACE addr_host ")"

addr_adl        ::= nstring
                    ;; Conserve le chemin à partir de la
                    ;; route-adress [RFC-822] si non vide

addr_host       ::= nstring
                    ;; NIL indique une syntaxe de
                    ;; groupe [RFC-822] sinon, contient
                    ;; le nom de domaine [RFC-822]
```

```
addr_mailbox    ::= nstring
                    ;; NIL(VIDE) indique la fin d'un groupe
                    ;; [RFC-822]; Si non-NIL et si addr_host
                    ;; est NIL, ça contient le nom de groupe
                    ;; [RFC-822].
                    ;; Sinon, contient la partie locale
                    ;; [RFC-822]

addr_name       ::= nstring
                    ;; contient la locution de la boîte
                    ;; aux lettres [RFC-822] si non-NIL

alpha           ::= "A" / "B" / "C" / "D" / "E" / "F" / "G" /
                    "H" / "I" / "J" / "K" / "L" / "M" / "N" /
                    "O" / "P" / "Q" / "R" / "S" / "T" / "U" /
                    "V" / "W" / "X" / "Y" / "Z" / "a" / "b" /
                    "c" / "d" / "e" / "f" / "g" / "h" / "i" /
                    "j" / "k" / "l" / "m" / "n" / "o" / "p" /
                    "q" / "r" / "s" / "t" / "u" / "v" / "w" /
                    "x" / "y" / "z"
                    ;; insensible à la casse

append          ::= "APPEND" SPACE mailbox [SPACE flag_list]
                    [SPACE date_time] SPACE literal

astring         ::= atom / string

atom            ::= 1*ATOM_CHAR

ATOM_CHAR       ::= <n'importe quel CHAR
                        excepté atom_specials>

atom_specials   ::= "(" / ")" / "{" / SPACE / CTL /
                    list_wildcards / quoted_specials


authenticate    ::= "AUTHENTICATE" SPACE auth_type
                     *(CRLF base64)

auth_type       ::= atom
                    ;; Défini par [IMAP-AUTH]

base64          ::= *(4base64_char) [base64_terminal]

base64_char     ::= alpha / digit / "+" / "/"

base64_terminal ::= (2base64_char "==") / (3base64_char "=")

body            ::= "(" body_type_1part / body_type_mpart ")"

body_extension  ::= nstring / number /
                    "(" 1#body_extension ")"
                    ;; Développement futur.
                    ;; Les implémentations du client
                    ;; DOIVENT (MUST) accepter les
                    ;; champs body_extension. Celle des
                    ;; serveurs NE DOIVENT PAS (MUST NOT)
                    ;; générer des champs body_extension
                    ;; excepté ceux défini par de futur
                    ;; standard ou révision standards-track à
                    ;; cette spécification.

body_ext_1part  ::= body_fld_md5 [SPACE body_fld_dsp
                    [SPACE body_fld_lang
                    [SPACE 1#body_extension]]]
                    ;; NE DOIT PAS (MUST NOT) être retourné
                    ;; à un fetch BODY non-extensible

body_ext_mpart  ::= body_fld_param
                    [SPACE body_fld_dsp SPACE body_fld_lang
                    [SPACE 1#body_extension]]
                    ;; NE DOIT PAS (MUST NOT) être retourné
```

```
                    ;; à un fetch BODY non-extensible

body_fields     ::= body_fld_param SPACE body_fld_id SPACE
                    body_fld_desc SPACE body_fld_enc SPACE
                    body_fld_octets

body_fld_desc   ::= nstring

body_fld_dsp    ::= "(" string SPACE body_fld_param ")" / nil

body_fld_enc    ::= (<"> ("7BIT" / "8BIT" / "BINARY" /
                    "BASE64"/ "QUOTED-PRINTABLE") <">)/
                    string

body_fld_id     ::= nstring

body_fld_lang   ::= nstring / "(" 1#string ")"

body_fld_lines  ::= number

body_fld_md5    ::= nstring

body_fld_octets ::= number

body_fld_param  ::= "(" 1#(string SPACE string) ")" / nil

body_type_1part ::= (body_type_basic / body_type_msg /
                     body_type_text) [SPACE body_ext_1part]

body_type_basic ::= media_basic SPACE body_fields
                    ;; Le sous-type NE DOIT PAS (MUST NOT)
                    ;; &ecirctre "RFC822"

body_type_mpart ::= 1*body SPACE media_subtype
                    [SPACE body_ext_mpart]

body_type_msg   ::= media_message SPACE body_fields SPACE
                    envelope SPACE body SPACE body_fld_lines

body_type_text  ::= media_text SPACE body_fields SPACE
                    body_fld_lines

capability      ::= "AUTH=" auth_type / atom
                    ;; Les nouvelles fonctionnalités DOIVENT
                    ;; (MUST) commencer par un "X" ou être
                    ;; enregistrée à l'IANA comme standard ou
                    ;; standards-track

capability_data ::= "CAPABILITY" SPACE [1#capability SPACE]
                    "IMAP4rev1" [SPACE 1#capability]
                    ;; Les serveurs IMAP4rev1 qui offre une
                    ;; fonctionnalité RFC 1730 doivent lister
                    ;; "IMAP4" comme première fonctionnalité.

CHAR            ::= <tout caractére 7-bit US-ASCII excepté
                    NUL, 0x01 - 0x07f>

CHAR8           ::= <tout octet sur 8-bit excepté
                     NUL, 0x01 - 0xff>

command         ::= tag SPACE (command_any / command_auth /
                    command_nonauth / command_select) CRLF
                    ;; Logique modale basée sur l'état

command_any     ::= "CAPABILITY" / "LOGOUT" / "NOOP" /
                    x_command
                    ;; Valide pour tout les états

command_auth    ::= append / create / delete / examine /
                    list / lsub / rename / select / status /
                    subscribe / unsubscribe
                    ;; Valide seulement en état authentifié
                    ;; ou sélectionné
```

```
command_nonauth ::= login / authenticate
                    ;; Valide seulement en état
                    ;; Non-Authentifié

command_select  ::= "CHECK" / "CLOSE" / "EXPUNGE" /
                     copy / fetch / store / uid / search
                    ;; Valide seulement état sélectionné

continue_req    ::= "+" SPACE (resp_text / base64)

copy            ::= "COPY" SPACE set SPACE mailbox

CR              ::= <ASCII CR, carriage return, 0x0D>

create          ::= "CREATE" SPACE mailbox
                    ;; L'utilisation de INBOX donne une
                    ;; erreur NO

CRLF            ::= CR LF

CTL             ::= <any ASCII control character and DEL,
                        0x00 - 0x1f, 0x7f>

date            ::= date_text / <"> date_text <">

date_day        ::= 1*2digit
                    ;; Jour du mois

date_day_fixed  ::= (SPACE digit) / 2digit
                    ;; Version avec un format fixe de
                    ;; date_day

date_month      ::= "Jan" / "Feb" / "Mar" / "Apr" / "May" /
                    "Jun" / "Jul" / "Aug" / "Sep" / "Oct" /
                    "Nov" / "Dec"

date_text       ::= date_day "-" date_month "-" date_year

date_year       ::= 4digit

date_time       ::= <"> date_day_fixed "-" date_month "-"
                    date_year SPACE time SPACE zone <">

delete          ::= "DELETE" SPACE mailbox
                    ;; L'utilisation de INBOX donne
                    ;; un erreur NO

digit           ::= "0" / digit_nz

digit_nz        ::= "1" / "2" / "3" / "4" / "5" / "6" / "7" /
                    "8" / "9"

envelope        ::= "(" env_date SPACE env_subject SPACE
                    env_from SPACE env_sender SPACE
                    env_reply_to SPACE env_to SPACE
                    env_cc SPACE env_bcc SPACE
                    env_in_reply_to SPACE env_message_id ")"

env_bcc         ::= "(" 1*address ")" / nil

env_cc          ::= "(" 1*address ")" / nil

env_date        ::= nstring

env_from        ::= "(" 1*address ")" / nil

env_in_reply_to ::= nstring

env_message_id  ::= nstring

env_reply_to    ::= "(" 1*address ")" / nil

env_sender      ::= "(" 1*address ")" / nil
```

```
env_subject     ::= nstring

env_to          ::= "(" 1*address ")" / nil

examine         ::= "EXAMINE" SPACE mailbox

fetch           ::= "FETCH" SPACE set SPACE ("ALL" / "FULL" /
                    "FAST" / fetch_att / "(" 1#fetch_att ")")

fetch_att       ::= "ENVELOPE" / "FLAGS" / "INTERNALDATE" /
                    "RFC822" [".HEADER" / ".SIZE" /
                    ".TEXT"] / "BODY" ["STRUCTURE"] /
                    "UID" / "BODY" [".PEEK"] section
                    ["<" number "." nz_number ">"]

flag            ::= "\Answered" / "\Flagged" / "\Deleted" /
                    "\Seen" / "\Draft" / flag_keyword /
                     flag_extension

flag_extension  ::= "\" atom
                    ;; Extension future. Les implémentations
                    ;; cliente DOIVENT (MUST) accepter les
                    ;; drapeaux flag_extension. Les
                    ;; implémentations serveur NE DOIVENT PAS
                    ;; (MUST NOT) générer des drapeaux
                    ;; flag_extension excepté ce qui est
                    ;; défini par de futurs standards ou des
                    ;; révisions standards-track de cette
                    ;; spécification.

flag_keyword    ::= atom

flag_list       ::= "(" #flag ")"

greeting        ::= "*" SPACE (resp_cond_auth /
                     resp_cond_bye) CRLF

header_fld_name ::= astring

header_list     ::= "(" 1#header_fld_name ")"

LF              ::= <ASCII LF, line feed, 0x0A>

list            ::= "LIST" SPACE mailbox SPACE list_mailbox

list_mailbox    ::= 1*(ATOM_CHAR / list_wildcards) / string

list_wildcards  ::= "%" / "*"

literal         ::= "{" number "}" CRLF *CHAR8
                    ;; Nombre representant le nombre d'octect
                    ;; de type CHAR8

login           ::= "LOGIN" SPACE userid SPACE password

lsub            ::= "LSUB" SPACE mailbox SPACE list_mailbox

mailbox         ::= "INBOX" / astring
                    ;; INBOX est insensible à la casse.
                    ;; Toutes les variantes de INBOX
                    ;; (par ex. "iNbOx") DOIVENT (MUST) être
                    ;; interprétées comme INBOX et non
                    ;; comme une chaîne de type "astring";
                    ;; référez vous à la section 5.1
                    ;; pour plus de détails sur
                    ;; la sémantique des noms de boîte aux
                    ;; lettres.

mailbox_data    ::=  "FLAGS" SPACE flag_list /
                     "LIST" SPACE mailbox_list /
                     "LSUB" SPACE mailbox_list /
                     "MAILBOX" SPACE text /
                     "SEARCH" [SPACE 1#nz_number] /
                     "STATUS" SPACE mailbox SPACE
```

```
                    "(" #<status_att number ")" /
                    number SPACE "EXISTS" /
                    number SPACE "RECENT"

mailbox_list    ::= "(" #("\Marked" / "\Noinferiors" /
                    "\Noselect" / "\Unmarked" /
                    flag_extension) ")" SPACE
                    (<"> QUOTED_CHAR <"> / nil) SPACE
                    mailbox

media_basic     ::= (<"> ("APPLICATION" / "AUDIO" / "IMAGE" /
                    "MESSAGE" / "VIDEO") <">) / string)
                    SPACE media_subtype
                    ;; Défini dans [MIME-IMT]

media_message   ::= <"> "MESSAGE" <"> SPACE <"> "RFC822" <">
                    ;; Défini dans [MIME-IMT]

media_subtype   ::= string
                    ;; Défini dans [MIME-IMT]

media_text      ::= <"> "TEXT" <"> SPACE media_subtype
                    ;; Défini dans [MIME-IMT]

message_data    ::= nz_number SPACE ("EXPUNGE" /
                                     ("FETCH" SPACE msg_att))

msg_att         ::= "(" 1#("ENVELOPE" SPACE envelope /
                    "FLAGS" SPACE
                    "(" #(flag / "\Recent") ")" /
                    "INTERNALDATE" SPACE date_time /
                    "RFC822" [".HEADER" / ".TEXT"] SPACE
                    nstring / "RFC822.SIZE" SPACE number /
                    "BODY" ["STRUCTURE"] SPACE body /
                    "BODY" section ["<" number ">"] SPACE
                    nstring / "UID" SPACE uniqueid) ")"

nil             ::= "NIL"

nstring         ::= string / nil

number          ::= 1*digit
                    ;; Entier non signé 32-bit
                    ;; (0 <= n < 4,294,967,296)

nz_number       ::= digit_nz *digit
                    ;; Entier non signé 32-bit
                    ;; en excluant zéro
                    ;; (0 < n < 4,294,967,296)

password        ::= astring

quoted          ::= <"> *QUOTED_CHAR <">

QUOTED_CHAR     ::= <tout TEXT_CHAR excepté quoted_specials>
                    / "\" quoted_specials

quoted_specials ::= <"> / "\"

rename          ::= "RENAME" SPACE mailbox SPACE mailbox
                    ;; L'utilisation de INBOX comme
                    ;; destination donne une erreur NO.

response        ::= *(continue_req / response_data)
                    response_done

response_data   ::= "*" SPACE (resp_cond_state /
                    resp_cond_bye / mailbox_data /
                    message_data /
                    capability_data) CRLF

response_done   ::= response_tagged / response_fatal

response_fatal  ::= "*" SPACE resp_cond_bye CRLF
```

```
                    ;; Le Serveur ferme les connexions
                    ;; immédiatement

response_tagged ::= tag SPACE resp_cond_state CRLF

resp_cond_auth  ::= ("OK" / "PREAUTH") SPACE resp_text
                    ;; Condition d'authentification

resp_cond_bye   ::= "BYE" SPACE resp_text

resp_cond_state ::= ("OK" / "NO" / "BAD") SPACE resp_text
                    ;; Condition de statut

resp_text       ::= ["[" resp_text_code "]" SPACE]
                    (text_mime2 / text)
                    ;; Le texte NE DEVRAIT PAS (SHOULD NOT)
                    ;; commencer par "[" ou par "="

resp_text_code  ::= "ALERT" / "PARSE" /
                    "PERMANENTFLAGS" SPACE
                     "(" #(flag / "\*") ")" /
                    "READ-ONLY" / "READ-WRITE" /
                    "TRYCREATE" /
                    "UIDVALIDITY" SPACE nz_number /
                    "UNSEEN" SPACE nz_number / atom
                    [SPACE 1*<tout TEXT_CHAR excepté "]">]

search          ::= "SEARCH" SPACE
                    ["CHARSET" SPACE astring SPACE]
                    1#search_key
                    ;; [CHARSET] DOIT (MUST) être enregistré
                    ;; par l'IANA

search_key      ::= "ALL" / "ANSWERED" / "BCC" SPACE
                     astring /
                    "BEFORE" SPACE date /
                    "BODY" SPACE astring /
                    "CC" SPACE astring / "DELETED" /
                    "FLAGGED" /
                    "FROM" SPACE astring /
                    "KEYWORD" SPACE flag_keyword /
                    "NEW" / "OLD" /
                    "ON" SPACE date / "RECENT" /
                    "SEEN" / "SINCE" SPACE date /
                    "SUBJECT" SPACE astring /
                    "TEXT" SPACE astring /
                    "TO" SPACE astring /
                    "UNANSWERED" / "UNDELETED" /
                    "UNFLAGGED" /
                    "UNKEYWORD" SPACE flag_keyword /
                    "UNSEEN" /
                    ;; Au dessous de cette ligne on est
                    ;; en [IMAP2].
                    "DRAFT" / "HEADER" SPACE
                    header_fld_name SPACE astring /
                    "LARGER" SPACE number / "NOT" SPACE
                    search_key / "OR" SPACE search_key SPACE
                    search_key / "SENTBEFORE" SPACE date /
                    "SENTON" SPACE date /
                    "SENTSINCE" SPACE date /
                    "SMALLER" SPACE number /
                    "UID" SPACE set /
                    "UNDRAFT" / set /
                    "(" 1#search_key ")"

section         ::= "[" [section_text /
                    (nz_number *["." nz_number]
                    ["." (section_text / "MIME")])] "]"

section_text    ::= "HEADER" / "HEADER.FIELDS" [".NOT"]
                    SPACE header_list / "TEXT"

select          ::= "SELECT" SPACE mailbox
```

```
sequence_num    ::= nz_number / "*"
                    ;; * est le plus grand nombre utilisé.
                    ;; Pour les numéros de séquence de
                    ;; message, c'est le nombre de message
                    ;; dans la boîte aux lettres. Pour des
                    ;; identifiants uniques, c'est
                    ;; l'identifiant unique du dernier
                    ;; message dans la boîte aux lettres.

set             ::= sequence_num /
                    (sequence_num ":" sequence_num) /
                    (set "," set)
                    ;; Identifie un jeu (série) de messages.
                    ;; Pour les numéros de séquences de
                    ;; message, on a des numéros consécutifs
                    ;; allant de 1 au nombre de message dans
                    ;; la boîtes aux lettres.
                    ;; Les virgules délimitent les nombres
                    ;; individuels, le deux points entre deux
                    ;; nombres, détermine l'intervalle
                    ;; compris entre ces deux nombres,
                    ;; tout en les y incluant.
                    ;; Exemple:
                    ;; 2,4:7,9,12:* est égal 2,4,5,6,7,9,12,
                    ;; 13,14,15 pour une boîte aux lettres de
                    ;; 15 messages.

SPACE           ::= <ASCII SP, space, 0x20>

status          ::= "STATUS" SPACE mailbox SPACE
                    "(" 1#status_att ")"

status_att      ::= "MESSAGES" / "RECENT" / "UIDNEXT" /
                    "UIDVALIDITY" / "UNSEEN"

store           ::= "STORE" SPACE set SPACE store_att_flags

store_att_flags ::= (["+" / "-"] "FLAGS" [".SILENT"]) SPACE
                    (flag_list / #flag)

string          ::= quoted / literal

subscribe       ::= "SUBSCRIBE" SPACE mailbox

tag             ::= 1*<tout ATOM_CHAR excepté "+">

text            ::= 1*TEXT_CHAR

text_mime2      ::= "=?" <charset> "?" <encoding> "?"
                    <encoded-text> "?="
                    ;; Syntaxe défini dans [MIME-HDRS]

TEXT_CHAR       ::= <tout CHAR excepté CR et LF>

time            ::= 2digit ":" 2digit ":" 2digit
                    ;; Heures minutes secondes

uid             ::= "UID" SPACE (copy / fetch / search /
                    store)
                    ;; Identifiants uniques utilisés à
                    ;; la place des numéros de séquence
                    ;; de message

uniqueid        ::= nz_number
                    ;; Strictement ascendant

unsubscribe     ::= "UNSUBSCRIBE" SPACE mailbox

userid          ::= astring

x_command       ::= "X" atom <arguments à une commande
                     expérimentale>
```

```
zone            ::= ("+" / "-") 4digit
                    ;; Valeur de hhmm sur quatre-digit avec
                    ;; un signe représentant les heures
                    ;; minutes ouest (west) de Greenwich
                    ;; (c.a.d., (la quantité que le temps
                    ;; qui diffère de Universal Time
                    ;; (temps universel). En soustrayant
                    ;; la zone de temps (time zone) du temps
                    ;; donné, cela donnera la form UT
                    ;; (ou TU). La zone Temps Uuniversel
                    ;; est "+0000.
```

- **10. Note de l'auteur**

  Ce document est l'amélioration et la réécriture de documents plus vieux, et remplace la spécification de protocole décrite dans les documents suivants : RFC 1730, le document non publié IMAP2bis.TXT, RFC 1176, et RFC 1064.

- **11. Considérations sécuritaires**

  Les transactions de protocole IMAP4rev1, y compris les données de courrier électronique, sont envoyées en clair sur le réseau à moins que la protection de l'intimité soit activée par l'intermédiaire de la commande AUTHENTICATE.

  Un message d'erreur pour la commande AUTHENTICATE qui échoue du fait de justification d'identité non correcte NE DEVRAIT PAS (SHOULD NOT) détailler pourquoi les accréditations sont incorrectes.

  L'utilisation de la commande LOGIN fait que l'on envoie les mots de passe en clair. Ceci peut être éviter en utilisant à la place la commande AUTHENTICATE.

  Aucun message d'erreur du serveur pour l'échec de la commande LOGIN NE DEVRAIT PAS (SHOULD NOT) préciser que le nom d'utilisateur, contrairement a ce qui doit ce faire pour le mot de passe, est incorrect.

  D'autres considérations de sécurités sont traitées dans la section parlant des commandes LOGIN et AUTHENTICATE.

- **12. Adresse de l'auteur**

Mark R. Crispin
Networks and Distributed Computing
University of Washington
4545 15th Aveneue NE
Seattle, WA 98105-4527
Téléphone: (206) 543-5762
EMail: MRC@CAC.Washington.EDU

- **Annexes**

- **A. Références**

[ACAP] Myers, J. "ACAP -- Application Configuration Access Protocol", Work in Progress.

[CHARSET] Reynolds, J., et J. Postel, "Assigned Numbers", STD 2, RFC 1700, USC/Information

Sciences Institute, Octobre 1994.

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

[IMAP-AUTH] Myers, J., "IMAP4 Authentication Mechanism", RFC 1731. Carnegie-Mellon University, Décembre 1994.

[IMAP-COMPAT] Crispin, M., "IMAP4 Compatibility with IMAP2bis", RFC 2061, University of Washington, Novembre 1996.

[IMAP-DISC] Austein, R., "Synchronization Operations for Disconnected IMAP4 Clients", Work in Progress.

[IMAP-HISTORICAL] Crispin, M. "IMAP4 Compatibility with IMAP2 and IMAP2bis", RFC 1732, University of Washington, Décembre 1994.

[IMAP-MODEL] Crispin, M., "Distributed Electronic Mail Models in IMAP4", RFC 1733, University of Washington, Décembre 1994.

[IMAP-OBSOLETE] Crispin, M., "Internet Message Access Protocol - Obsolete Syntax", RFC 2062, University of Washington, Novembre 1996.

[IMAP2] Crispin, M., "Interactive Mail Access Protocol - Version 2", RFC 1176, University of Washington, Août 1990.

[LANGUAGE-TAGS] Alvestrand, H., "Tags for the Identification of Languages", RFC 1766, Mars 1995.

[MD5] Myers, J., et M. Rose, "The Content-MD5 Header Field", RFC 1864, Octobre 1995.

[MIME-IMB] Freed, N., et N. Borenstein, "MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies", RFC 2045, Novembre 1996.

[MIME-IMT] Freed, N., et N. Borenstein, "MIME (Multipurpose Internet Mail Extensions) Part Two: Media Types", RFC 2046, Novembre 1996.

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

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

[SMTP] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, USC/Information Sciences Institute, Août 1982.

[UTF-7] Goldsmith, D., et Davis, M., "UTF-7: A Mail-Safe Transformation Format of Unicode", RFC 1642, Juillet 1994.

- **B. Changement par rapport à la RFC 1730**

1) La commande STATUS a été ajoutée.

2) On a clarifié la syntaxe de forme où le constructeur "#" ne peut jamais faire référence à plusieurs espaces.

3) La syntaxe obsolète a été déplacée vers un document séparé.

4) La commande PARTIAL est devenue obsolète.

5) Les attributs de fetch : RFC822.HEADER.LINES, RFC822.HEADER.LINES.NOT, RFC822.PEEK, et RFC822.TEXT.PEEK sont devenu obsolète.

6) Le suffixe "<" origin "." size ">" pour les attributs de texte a été ajouté.

7) Les identifiants des morceaux HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, et TEXT ont été ajouté.

8) Le support pour Content-Disposition et Content-Language a été ajouté.

9) La restriction concernant sur la récupération des parties MULTIPART imbriquées a été enlevée.

10) Le numéro 0 de partie de corps a été mis obsolète.

11) Les authentifiants supporté par le serveur sont maintenant identifié par les capacités.

12) La capacité qui identifie ce protocole est maintenant appelé "IMAP4rev1". Un serveur qui fournie un rétrosupport de la RFC 1730 DEVRA (SHOULD) émettre la capacité "IMAP4" en plus de "IMAP4rev1" dans sa réponse à CAPABILITY. Parce que la RFC1730 nécessite que "IMAP4" apparaisse comme première de ces capacités, elle doit (MUST) être listée en premier dans la réponse.

13) Une description de la convention d'espace de nom de boîte aux lettres a été ajouté.

14) Une description de la convention internationale de nom de boîte aux lettres a été ajoutée.

15) Les éléments de statut UID-VALIDITY et UID-INEXT sont appelé maintenant UIDNEXT et UIDVALIDITY. Ceci a été changé du fait du travail qui a été fourni et non de RFC-1730 (sic: traduction très approximative)

16) Une clarification a été ajoutée en ce qui concerne l'argument de nom de boîte aux lettres null pour une commande LIST, elle retourne une réponse LIST avec le délimiteur de hiérarchie et la racine de l'argument de référence.

17) La définition des termes tels que "DOIT", "DEVRAIT" et "NE DOIT PAS" ("MUST", "SHOULD", et "MUST NOT") a été ajoutée.

18) On a jouté 'une section qui défini les attributs de message et qui détaille complètement la sémantique des numéros de séquence, de message UID, et drapeaux.

19) On a ajouté un éclaircissement détaillant les circonstances pour lesquelles un client peut envoyer de multiples commandes sans attendre une réponse, et les circonstances pour lesquelles des ambiguïtés pourraient potentiellement en résulter.

20) on a ajouté une recommandation en ce qui concerne le comportement du serveur pour DELETE et RENAME quand des noms hiérarchiquement inférieurs à un nom donné existe.

21) On a ajouté un éclaircissement : Une boîte aux lettres ne peut être unilatéralement desenregistré par le serveur, même si ce nom de boîte aux lettres n'existe plus depuis longtemps.

22) On a ajouté un éclaircissement en ce qui concerne LIST. LIST devrait retourner ses résultats rapidement sans délais indus.

23) On a ajouté un éclaircissement pour l'argument date_time de APPEND qui fixe la date interne du message.

24) Un éclaircissement a été ajouté en ce qui concerne le comportement de APPEND quand la boîte aux lettres cible est la boîte aux lettres actuellement sélectionnée.

25) On a ajouté une précision selon laquelle les changements externes de drapeaux devront être (should be) toujours annoncé par un fetch de type non-marqué même si la commande en cours est un STORE avec le suffixe ".SILENT".

26) On a ajouté une clarification disant que COPY fait un ajout à la boîte aux lettres cible.

27) On a ajouté le code réponse de NEWNAME.

28) On a réécrit la description de la réponse BYE de type non marqué afin d'éclaircir sa sémantique.

29) On a changé la référence de MD5 pour qu'elle se réfère à la RFC adéquate.

30) On a clarifié les règles de syntaxe formelle qui se superposent, et qui lors de ces cas de superposition, c'est la règle qui arrive en premier qui remplace les autres.

31) On a corrigé la définition de body_fld_param.

32) On a donné une syntaxe plus formelle pour capability_data.

33) On a donné des éclaircissements en ce qui concerne toutes les variantes de "INBOX" qui doivent être interprétées comme INBOX.

34) On a donné des éclaircissements en ce qui concerne le texte visible par un humain dans un texte de type resp_text : Il ne devra (should not) pas commencer par "[" ou "=".

35) On a changé les références MIME en documents Draft Standard (sic: traduction très approximative).

36) On a clarifié la sémantique de \Recents.

37) On a des exemples supplémentaires.

- **C. Index des mots clé**

(Remarque hors RFC: L'index des mots clé est moins fidèle à l'original que ne l'est le reste du document : les références aux pages ont été transformées en référence à des sections, et les arguments de certains mots clé ont été supprimés pour des questions de lisibilité.)

Fin de traduction.