

Groupe de travail Réseau
Request for Comments : 3501
 RFC rendue obsolète : 2060
 Catégorie : En cours de normalisation

M. Crispin
 University of Washington
 mars 2003
 Traduction Claude Brière de L'Isle

Protocole d'accès aux messages Internet (IMAP) - Version 4rev1

Statut du présent mémoire

Le présent document spécifie un protocole de normalisation Internet pour la communauté Internet, et appelle à discussion et suggestions en vue de son amélioration. Prière de se rapporter à l'édition en cours des "Internet Official Protocol Standards" (normes officielles du protocole Internet) (STD 1) pour connaître l'état de la normalisation et le statut du présent protocole. La distribution du présent mémo n'est soumise à aucune restriction.

(Le présent texte incorpore les corrections indiquées par le rapport d'erreurs qui porte l'identifiant 261)

Déclaration de copyright

Copyright (C) The Internet Society (2003). Tous droits réservés

Résumé

Le protocole d'accès aux messages (IMAP4rev1, *Internet Internet Message Access Protocol, version 4rev1*) permet à un client d'accéder aux messages de messagerie électronique sur un serveur et de les manipuler. IMAP4rev1 permet la manipulation de boîtes aux lettres (dossiers de messages distants) d'une façon qui est fonctionnellement équivalente à celle des dossiers locaux. IMAP4rev1 fournit aussi la capacité pour un client hors ligne de se resynchroniser avec le serveur.

IMAP4rev1 inclut des opérations de création, suppression, et renommage de boîtes aux lettres, recherche de nouveaux messages, suppression permanente de messages, établissement et suppression de fanions, analyse grammaticale selon les RFC 2822 et RFC 2045, recherche, et assemblage sélectif d'attributs de messages, textes, et de portions de ceux-ci. L'accès aux messages dans IMAP4rev1 est effectué par l'utilisation de numéros. Ces numéros sont des numéros de séquence de message ou des identifiants univoques.

IMAP4rev1 prend en charge un seul serveur. Un mécanisme d'accès aux informations de configuration pour prendre en charge plusieurs serveurs IMAP4rev1 est exposé dans la RFC 2244.

IMAP4rev1 ne spécifie pas un moyen pour envoyer de la messagerie ; cette fonction est traitée par un protocole de transfert de messagerie tel que celui de la RFC 2821.

Table des matières

Protocole d'accès aux messages Internet (IMAP) - Version 4rev1.....	1
1. Comment lire ce document.....	2
1.1 Organisation du document.....	2
1.2 Conventions utilisées dans le document.....	2
1.3 Notes particulières pour les développeurs.....	3
2. Généralité sur le protocole.....	3
2.1 Niveau liaison.....	3
2.2 Commandes et réponses.....	3
2.3 Attributs de message.....	5
2.4 Textes de message.....	7
3. État et diagramme de flux.....	7
3.1 État Non authentifié.....	7
3.2 État Authentifié.....	7
3.3 État Choisi.....	8
3.4 État Fin de connexion.....	8
4. Formats de données.....	9
4.1 Atome.....	9
4.2 Nombre.....	9
4.3 Chaîne.....	9
4.4 Liste entre parenthèses.....	9

4.5 NIL.....	10
5. Considérations pour le fonctionnement.....	10
5.1 Dénomination de boîte aux lettres.....	10
5.2 Taille de boîte aux lettres et mises à jour d'état de message.....	12
5.3 Réponse quand aucune commande n'est en cours.....	12
5.4 Temporisateur d'auto déconnexion.....	12
5.5 Plusieurs commandes en cours.....	12
6. Commandes du client.....	13
6.1 Commandes du client – tous états.....	13
6.2 Commandes du client – état Non authentifié.....	15
6.3 Commandes du client – état Authentifié.....	18
6.4 Commandes du client – état Choisi.....	27
6.5 Commandes du client – Expérimental/Expansion.....	36
7. Réponses du serveur.....	36
7.1 Réponses du serveur – Réponses d'état.....	37
7.2 Réponses du serveur – État du serveur et de la boîte aux lettres.....	39
7.3 Réponses du serveur – Taille de boîte au lettres.....	41
7.4 Réponses du serveur – état du message.....	42
7.5 Réponses du serveur – Demande de continuation de commande.....	46
8. Exemple de connexion IMAP4rev1.....	46
9. Syntaxe formelle.....	47
10. Note de l'auteur.....	54
11. Considérations pour la sécurité.....	54
11.1 Considérations pour la sécurité sur STARTTLS.....	54
11.2 Autres considérations pour la sécurité.....	54
12. Considérations relatives à l'IANA.....	55
Appendice A Références normatives.....	55
A.1 Références informatives.....	56
Appendice B Changements par rapport à la RFC 2060.....	57
Appendice C. Index des mots clés.....	59
Déclaration de copyright.....	61

1. Comment lire ce document

1.1 Organisation du document

Le présent document est écrit du point de vue de la mise en œuvre d'un client ou serveur IMAP4rev1. Au delà des généralités sur le protocole de la section 2, il n'est pas optimisé pour quelqu'un qui essaierait de comprendre le fonctionnement du protocole. Les sections 3 à 5 fournissent le contexte général et les définitions avec lesquels fonctionne IMAP4rev1.

Les sections 6, 7, et 9 décrivent respectivement les commandes, les réponses et la syntaxe IMAP. Les relations entre celles-ci sont telles qu'il est presque impossible d'en comprendre une séparément des autres. En particulier, ne pas essayer de déduire la syntaxe d'une commande à partir de la seule section sur les commandes ; se référer plutôt à la section Syntaxe formelle.

1.2 Conventions utilisées dans le document

Les "conventions" sont des principes ou procédures de base. Les conventions du document sont notées dans cette section.

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

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "PEUT", et "FACULTATIF" dans ce document sont à interpréter comme décrit dans [RFC2119].

Le mot "peut" est utilisé pour se référer à une circonstance ou situation possible, par opposition à une facilité facultative du protocole.

"Usager" est utilisé pour se référer à un utilisateur humain, alors que "client" se réfère au logiciel utilisé par l'utilisateur.

"Connexion" se réfère à la séquence entière d'interaction client/serveur à partir de l'établissement initial de la connexion réseau jusqu'à ce qu'il y soit mis un terme.

"Session" se réfère à la séquence d'interaction client/serveur depuis le moment du choix d'une boîte aux lettres (commande SELECT ou EXAMINE) jusqu'au moment où cette sélection se termine (commandes SELECT ou EXAMINE d'une autre boîte aux lettres, commande CLOSE, ou terminaison de connexion).

Les caractères sont en US-ASCII à 7 bits sauf spécification contraire. Les autres jeux de caractères sont indiqués en utilisant un "CHARSET", comme décrit dans [RFC2046] et défini dans [CHARSET]. Les CHARSET ont une sémantique supplémentaire importante en plus de la définition de jeu de caractères ; se référer à ces documents pour des précisions.

Il y a plusieurs conventions de protocole dans IMAP. Elles se réfèrent à des aspects de la spécification qui ne font pas strictement partie du protocole IMAP, mais reflètent une pratique généralement acceptée. Les mises en œuvre doivent connaître ces conventions, et éviter les conflits qu'elles appliquent ou non la convention. Par exemple, "&" ne doit pas être utilisé comme délimiteur hiérarchique car il entre en conflit avec la convention internationale de dénomination des boîtes aux lettres, et les autres utilisations de "&" dans les noms de boîtes aux lettres sont aussi impactés.

1.3 Notes particulières pour les développeurs

Les développeurs du protocole IMAP sont vivement encouragés à lire le document des recommandations de mise en œuvre de IMAP [IMAP-IMPL] en conjonction avec le présent document, pour mieux comprendre les complications de ce protocole et comment mieux construire un produit interopérable.

IMAP4rev1 est conçu pour être rétro-compatible avec les protocoles IMAP2 [RFC1176] et IMAP2bis (non publiée). IMAP4rev1 est largement compatible avec le protocole IMAP4 décrit dans la RFC 1730 ; à l'exception de certaines facilités ajoutées dans la RFC 1730 qui se sont révélées problématiques et ont été retirées ultérieurement. Dans le cours de l'évolution de IMAP4rev1, certains aspects des protocoles antérieurs sont devenus obsolètes. Les commandes, réponses, et formats de données obsolètes qu'une mise en œuvre IMAP4rev1 peut rencontrer lorsqu'elle est utilisée avec une mise en œuvre antérieure sont décrites dans [RFC2062].

Les autres questions de compatibilité avec IMAP2bis, variante la plus courante du protocole antérieur, sont exposées dans la [RFC 2061]. Un exposé complet des questions de compatibilité avec ces variantes rares (et présumées disparues) de IMAP2 figure dans la [RFC 1732] ; ce document est d'un intérêt principalement historique.

IMAP a été développé à l'origine pour l'ancienne norme [RFC-822], et par conséquent, plusieurs éléments d'assemblage dans IMAP incorporent "RFC822" dans leur nom. À l'exception de RFC822.SIZE, il y a des remplaçants plus modernes ; par exemple, la version moderne de RFC822.HEADER est BODY.PEEK[HEADER]. Dans tous les cas, "RFC822" devait être interprété comme une référence à la norme [RFC-2822] mise à jour.

2. Généralité sur le protocole

2.1 Niveau liaison

Le protocole IMAP4rev1 suppose un flux de données fiable tel que celui fourni par TCP. Lorsque TCP est utilisé, un serveur IMAP4rev1 écoute sur l'accès 143.

2.2 Commandes et réponses

Une connexion IMAP4rev1 consiste en l'établissement d'une connexion réseau client/serveur, en accueil initial de la part du serveur, et des interactions client/serveur. Ces interactions client/serveur consistent en une commande du client, en données du serveur, et en une réponse d'achèvement du résultat du serveur.

Toutes les interactions transmises par le client et le serveur sont sous la forme de lignes, c'est à dire, de chaînes qui se terminent par un CRLF. Le receveur de protocole d'un client ou serveur IMAP4rev1 lit une ligne, ou lit une séquence d'un nombre connu d'octets suivie par une ligne.

2.2.1 Envoi de protocole client et réception de protocole serveur

La commande du client commence une opération. Chaque commande du client est précédée d'un identifiant (normalement une courte chaîne alphanumérique, par exemple, A0001, A0002, etc.) appelée une "étiquette". Pour chaque commande, le client génère une étiquette différente.

Les clients DOIVENT suivre strictement la syntaxe exprimée dans la présente spécification. C'est une erreur de syntaxe d'envoyer une commande avec des espaces ou arguments manquants ou en trop.

Il y a deux cas dans lesquels une ligne provenant du client ne représente pas une commande complète. Dans un cas, un argument de commande est cité avec un compte d'octets (voir la description du littéral dans Chaîne sous Formats de données) ; dans l'autre cas, les arguments de commande exigent un retour de la part du serveur (voir la commande AUTHENTICATE). Dans tous les cas, le serveur envoie une réponse à la demande de continuation de commande si il est prêt pour les octets (si c'est approprié) et le reste de la commande. Cette réponse est précédée du jeton "+".

Note : Si à la place, le serveur détecte une erreur dans la commande, il envoie une réponse d'achèvement BAD avec une étiquette correspondant à la commande (comme décrit ci-dessous) pour rejeter la commande et empêcher le client d'en envoyer plus.

Il est aussi possible au serveur d'envoyer une réponse d'achèvement pour quelque autre commande (si plusieurs commandes sont en cours), ou des données non étiquetées. Dans tous les cas, la demande de continuation de commande est toujours en instance ; le client effectue l'action appropriée pour la réponse, et lit une autre réponse du serveur. Dans tous les cas, le client DOIT envoyer une commande complète (y compris de recevoir toutes les réponses de demande de continuation de commande et les continuations de commande 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 du serveur et une réponse de résultat d'achèvement de la commande du serveur.

2.2.2 Envoi de protocole serveur et receveur de protocole client

Les données transmises par le serveur au client et les réponses d'état qui n'indiquent pas l'achèvement de la commande sont précédées du jeton "*", et sont appelées des réponses non étiquetées.

Les données de serveur PEUVENT être envoyées comme résultat d'une commande du client, ou PEUVENT être envoyées de façon unilatérale par le serveur. Il n'y a pas de différence syntaxique entre les données du serveur qui résultent d'une commande spécifique et les données du serveur qui ont été envoyées de façon unilatérale.

La réponse d'achèvement du résultat du serveur indique le succès ou l'échec de l'opération. Elle est étiquetée avec la même étiquette que la commande du client qui a commencé l'opération. Et donc, si plus d'une commande est en cours, l'étiquette dans une réponse d'achèvement de serveur identifie la commande à laquelle la réponse s'applique. Il y a trois réponses d'achèvement du serveur possibles : OK (indique le succès), NO (indique l'échec), ou BAD (indique une erreur de protocole telle qu'une commande non reconnue ou une erreur de syntaxe de la commande).

Les serveurs DEVRAIENT mettre strictement en application la syntaxe décrite dans la présente spécification. Toute commande du client qui comporte une erreur de la syntaxe du protocole, y compris (sans s'y limiter) des espaces ou arguments manquants ou en trop, DEVRAIT être rejetée, et le client DEVRAIT recevoir une réponse d'achèvement du serveur de BAD.

Le receveur de protocole d'un client IMAP4rev1 lit une ligne de réponse provenant du serveur. Il entreprend alors l'action sur la réponse, sur la base du premier jeton de la réponse, qui peut être une étiquette, une "*", ou un "+".

Un client DOIT être prêt à accepter à tout moment toute réponse de serveur.

Cela inclut les données du serveur qui n'étaient pas demandées. Les données de serveur DEVRAIENT être enregistrées, de sorte que le client puisse se reporter à sa copie enregistrée plutôt que d'envoyer une commande au serveur pour demander les données. Dans le cas de certaines données du serveur, les données DOIVENT être enregistrées.

Cette question est exposée plus en détail dans la section Réponses du serveur.

2.3 Attributs de message

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

2.3.1 Numéros de message

Dans IMAP4rev1, on accède aux messages par un ou deux numéros : l'identifiant univoque ou le numéro de séquence du message.

2.3.1.1 Attribut de message Identifiant univoque (UID)

Une valeur de 32 bits non signés est allouée à chaque message, et lorsqu'elle est utilisée avec la valeur de validité d'identifiant univoque (voir ci-dessous) elle forme une valeur de 64 bits qui NE DOIT PAS se référer à un autre message dans la boîte aux lettres ou dans une boîte aux lettres suivante avec le même nom, et cela sans limite dans le temps. Les identifiants uniques sont alloués d'une façon strictement ascendante dans la boîte aux lettres ; lorsque chaque message est ajouté à la boîte aux lettres, il lui est alloué un UID supérieur à celui du ou des messages qui ont été ajoutés précédemment. À la différence des numéros de séquence de message, les identifiants univoques ne sont pas nécessairement contigus.

L'identifiant univoque d'un message NE DOIT PAS changer durant la session, et NE DEVRAIT PAS changer entre les sessions. Tout changement d'identifiant univoque entre les sessions DOIT être détectable en utilisant le mécanisme UIDVALIDITY exposé plus loin. Des identifiants univoques persistents sont exigés pour qu'un client resynchronise son état par rapport à une session précédente avec le serveur (par exemple, des clients déconnectés ou en accès hors ligne) ; ceci est exposé plus en détails dans [IMAP-DISC].

Deux valeurs de 32 bits non signées sont associées à chaque boîte aux lettres, qui aident au traitement de l'identifiant univoque : la prochaine valeur d'identifiant univoque (UIDNEXT) et la valeur de validité d'identifiant univoque (UIDVALIDITY).

La prochaine valeur d'identifiant univoque est la valeur prédite qui sera allouée à un nouveau message dans la boîte aux lettres. Sauf si la validité de l'identifiant univoque change aussi (voir ci-dessous), la prochaine valeur d'identifiant univoque DOIT avoir les deux caractéristiques suivantes. D'abord, la prochaine valeur d'identifiant univoque NE DOIT PAS changer à moins que de nouveaux messages ne soient ajoutés à la boîte aux lettres ; et ensuite, la prochaine valeur d'identifiant univoque DOIT changer chaque fois que de nouveaux messages sont ajoutés à la boîte aux lettres, même si ces nouveaux messages sont ultérieurement effacés.

Note : La prochaine valeur d'identifiant univoque est destinée à fournir un moyen pour qu'un client détermine si des messages ont été livrés à la boîte aux lettres depuis la dernière fois qu'il a vérifié cette valeur. Elle n'est pas destinée à fournir la moindre garantie qu'un message aura cet identifiant univoque. Un client peut seulement supposer, au moment où il obtient la prochaine valeur d'identifiant univoque, que les messages qui arrivent après ce moment auront un UID supérieur ou égal à cette valeur.

La valeur de validité d'identifiant est envoyée dans un code de réponse UIDVALIDITY au sein d'une réponse OK non étiquetée au moment du choix de boîte aux lettres. Si les identifiants univoques provenant d'une session antérieure ne réussissent pas à persister dans cette session, la valeur de validité d'identifiant univoque DOIT être supérieure à celle utilisée dans la session antérieure.

Note : Dans l'idéal, les identifiants univoques DEVRAIENT persister tout le temps. Bien que la présente spécification reconnaisse que l'incapacité à persister soit inévitable dans certains environnements de serveurs, elle RECOMMANDE FORTEMENT que les techniques de mise en œuvre de mémorisation de message évitent ce problème. Par exemple :

- 1) Les identifiants uniques DOIVENT être strictement ascendants à tout moment dans la boîte aux lettres. Si la mémorisation physique de message est réordonnée par un agent non IMAP, cela exige que les identifiants univoques dans la boîte aux lettres soient régénérés, car les anciens identifiants univoques ne sont plus strictement ascendants par suite de la réorganisation.
- 2) Si la mémorisation de message n'a pas de mécanisme pour mémoriser les identifiants univoques, elle doit régénérer les identifiants univoques à chaque session, et chaque session doit avoir une valeur unique UIDVALIDITY.
- 3) Si la boîte aux lettres est supprimée et qu'une nouvelle boîte aux lettres est créée avec le même nom à une date ultérieure, le serveur doit soit garder trace des identifiants univoques de l'instance précédente de boîte aux lettres, soit allouer une nouvelle valeur UIDVALIDITY à la nouvelle instance de la boîte aux lettres. Une bonne valeur de

UIDVALIDITY à utiliser dans ce cas est une représentation sur 32 bits de la date/heure de création de la boîte aux lettres. Il est correct d'utiliser une constante telle que 1, mais seulement si on a la garantie que ces identifiants univoques ne seront jamais réutilisés, même dans le cas de la suppression d'une boîte aux lettres (ou d'un changement de dénomination) et d'une création d'une nouvelle boîte aux lettres du même nom un peu plus tard.

- 4) La combinaison du nom, UIDVALIDITY, et UID d'une boîte aux lettres doit se référer pour toujours à un seul message immuable sur ce serveur. En particulier, la taille de date interne, [RFC-2822], l'enveloppe, la structure du corps, et les textes de message (éléments d'apports de données RFC822, RFC822.HEADER, RFC822.TEXT, et tous les BODY[...]) ne doivent jamais changer. Cela n'inclut pas les numéros de message, ni les attributs qui peuvent être établis par une commande STORE (par exemple, FLAGS).

2.3.1.2 Attribut de message Numéro de séquence de message

C'est une position relative à partir de 1 au nombre de messages dans la boîte aux lettres. Cette position DOIT être ordonnée par identifiant univoque ascendant. Lorsque chaque nouveau message est ajouté, il lui est alloué un numéro de séquence du message qui est supérieur de 1 au nombre de messages dans la boîte aux lettres avant l'ajout du nouveau message.

Les numéros de séquence de message peuvent être réalloués durant la session. Par exemple, lorsque un message est retiré de façon permanente (supprimé) de la boîte aux lettres, le numéro de séquence du message pour tous les messages suivants est décrémenté. Le nombre de messages dans la boîte aux lettres est aussi décrémenté. De même, un nouveau message peut se voir allouer un numéro de séquence de message qui a été détenu une autre fois par quelque autre message avant une suppression.

En plus de l'accès aux messages par la position relative dans la boîte aux lettres, les numéros de séquence du message peuvent être utilisés dans des calculs mathématiques. Par exemple, si un "11 EXISTS" non étiqueté est reçu, et si un "8 EXISTS" non étiqueté a été reçu précédemment, trois nouveaux messages sont arrivés avec les numéros de séquence de message de 9, 10, et 11. Autre exemple, si le message 287 dans une boîte aux lettres de 523 messages a l'UID 12345, il y a exactement 286 messages qui ont des UID inférieurs et 236 messages qui ont des UID supérieurs.

2.3.2 Attribut de message Fanions

Liste de zéro ou plus jetons nommés associés au message. Un fanion est mis par son ajout à cette liste, et il est supprimé par son retrait. Il y a deux types de fanions dans IMAP4rev1. Un fanion de l'un ou l'autre type peut être permanent ou seulement pour la session.

Un fanion système est un nom de fanion qui est pré-défini dans la présente spécification. Tous les fanions système commencent par "\". Certains fanions système (\Supprimé et \Lu) ont une sémantique particulière décrite plus loin. Les fanions système définis actuellement sont :

\Lu : le message a été lu

\Répondu : il a été répondu au message

\Fanion : le message est "étiqueté" pour attention/urgence particulière

\Supprimé : le message est "supprimé" pour effacement par un EXPUNGE ultérieur

\Brouillon : la composition du message n'est pas achevée (marqué comme projet).

\Récemment : le message est arrivé "récemment" dans la boîte aux lettres. Cette session est la première session qui a notification de ce message ; si la session est en lecture-écriture, les sessions ultérieures ne verront pas \Récemment établi pour ce message. Ce fanion ne peut pas être modifié par le client.

Si il n'est pas possible de déterminer si cette session est ou non la première session à recevoir notification d'un message, ce message DEVRAIT alors être considéré comme récemment.

Si plusieurs connexions ont simultanément la même boîte aux lettres sélectionnée, on ne peut pas définir laquelle de ces connexions verra les messages nouvellement arrivés avec \Récemment mis et laquelle les verra sans \Récemment mis.

Un mot clé est défini par la mise en œuvre de serveur. Les mots clé ne commencent pas par "\". Les serveurs PEUVENT permettre au client de définir de nouveaux mots clés dans la boîte aux lettres (voir la description du code de réponse PERMANENTFLAGS pour des informations complémentaires).

Un fanion peut être permanent ou seulement pour une session fanion par fanion. Les fanions permanents sont ceux que le client peut ajouter ou retirer de façon permanente des fanions de message ; c'est-à-dire que des sessions concurrentes et successives connaîtront des changements des fanions permanents. Les changements des fanions de session ne sont valides que dans cette session.

Note : Le fanion système \Récent est un cas particulier de fanion de session. \Récent ne peut pas être utilisé comme argument dans une commande STORE ou APPEND, et ne peut donc pas être changé du tout.

2.3.3 Attribut de message Date interne

C'est la date et l'heure interne du message sur le serveur. Ce n'est pas la date et l'heure dans l'en-tête de la [RFC2822], mais plutôt une date et heure qui reflète le moment où le message a été reçu. Dans le cas de messages délivrés via [SMTP], ce DEVRAIT être la date et l'heure de la livraison finale du message telle que définie par [SMTP]. Dans le cas de messages délivrés par la commande COPY de IMAP4rev1, ce DEVRAIT être la date et l'heure interne du message source. Dans le cas de messages délivrés par la commande IMAP4rev1 APPEND, ce DEVRAIT être la date et l'heure spécifiée dans la description de la commande APPEND. Tous les autres cas sont définis par la mise en œuvre.

2.3.4 Attribut de message Taille [RFC-2822]

C'est le nombre d'octets dans le message, comme exprimé dans le format de la [RFC2822].

2.3.5 Attribut de message Structure d'enveloppe

C'est une représentation analysée de l'en-tête [RFC2822] du message. Noter que la structure d'enveloppe IMAP n'est pas la même qu'une enveloppe [SMTP].

2.3.6 Attribut de message Structure de corps

C'est une représentation analysée des informations de structure du corps MIME-IMB [RFC2045] du message.

2.4 Textes de message

En plus d'être capable d'aller chercher le texte [RFC2822] complet d'un message, IMAP4rev1 permet d'aller chercher des portions du texte de message complet. En particulier, il est possible d'aller chercher l'en-tête du message [RFC2822], le corps du message [RFC2822], une partie de corps MIME-IMB, ou un en-tête MIME-IMB.

3. État et diagramme de flux

Une fois que la connexion entre client et serveur est établie, une connexion IMAP4rev1 est dans un des quatre états suivants. L'état initial est identifié dans l'accueil du serveur. La plupart des commandes ne sont valides que dans certains états. C'est une erreur de protocole de la part du client de tenter une commande alors que la connexion est dans un état inapproprié, et le serveur répondra par un résultat d'achèvement de commande BAD ou NO (selon la mise en œuvre de serveur).

3.1 État Non authentifié

Dans l'état Non authentifié, le client DOIT fournir les accreditifs d'authentification avant que la plupart des commandes ne soient permises. Cet état est atteint lorsque une connexion commence, sauf si la connexion a été préauthenticée.

3.2 État Authentifié

Dans l'état Authentifié, le client est authentifié et DOIT choisir une boîte aux lettres pour y accéder avant que les commandes qui affectent les messages ne soient permises. On entre dans cet état lorsque commence une connexion pré-

authentifiée, lorsque des accréditifs d'authentification acceptables ont été fournis, après une erreur dans le choix d'une boîte aux lettres, ou après une commande CLOSE réussie.

3.3 État Choisi

Dans un état Choisi, une boîte aux lettres a été choisie pour y accéder. On entre dans cet état lorsque une boîte aux lettres a été choisie avec succès.

3.4 État Fin de connexion

Dans l'état Fin de connexion, la connexion est terminée. On peut entrer dans cet état par suite d'une demande de client (via la commande LOGOUT) ou par une action unilatérale de la part du client ou du serveur.

Si le client demande l'état Fin de connexion, le serveur DOIT envoyer une réponse BYE non étiquetée et une réponse OK étiquetée à la commande LOGOUT avant que le serveur close la connexion ; et le client DOIT lire la réponse OK étiquetée à la commande LOGOUT avant que le client ne close la connexion.

Un serveur NE DOIT PAS clore unilatéralement la connexion sans envoyer une réponse non étiquetée BYE qui contient la cause de ce qu'il vient de faire. Un client NE DEVRAIT PAS clore unilatéralement la connexion, et DEVRAIT plutôt émettre une commande LOGOUT. Si le serveur détecte que le client clôt unilatéralement la connexion, le serveur PEUT omettre la réponse BYE non étiquetée et simplement clore sa connexion.

```

+-----+
| connexion établie |
+-----+
      ||
      \/\
+-----+
|          accueil du serveur          |
+-----+
      || (1)          || (2)          || (3)
      \/\            ||              ||
+-----+            ||              ||
| Non authentifié |      ||              ||
+-----+            ||              ||
      || (7)      || (4)          ||              || | |
      ||          \/\            \/\            ||
      ||          +-----+            ||
      ||          | Authentifié |<=++      ||
      ||          +-----+            ||
      ||          || (7)      || (5)      || (6)      ||
      ||          ||          \/\            ||              ||
      ||          ||          +-----+      ||              ||
      ||          ||          | Choix |==++      ||              ||
      ||          ||          +-----+      ||              ||
      ||          ||          || (7)          ||              ||
      \/\          \/\          \/\            \/\            ||
+-----+
|          Fin de connexion          |
+-----+
      ||
      \/\
+-----+
| les deux côtés ferment la connexion |
+-----+

```

- (1) connexion sans pré-authentification (accueil OK)
- (2) connexion pré-authentifiée (accueil PREAUTH)
- (3) connexion rejetée (accueil BYE)
- (4) commande LOGIN ou AUTHENTICATE réussie

- (5) commande SELECT ou EXAMINE réussie
- (6) commande CLOSE, ou échec de commande SELECT ou EXAMINE
- (7) commande LOGOUT, fermeture du serveur, ou connection close

4. Formats de données

IMAP4rev1 utilise des commandes et réponses textuelles. Les données en IMAP4rev1 peuvent être d'une parmi plusieurs formes : atome, nombre, chaîne, liste entre parenthèses, ou NIL. Noter qu'un élément de données particulier peut prendre plus d'une forme ; par exemple, un élément de données défini comme utilisant la syntaxe "astring" peut être un atome ou une chaîne.

4.1 Atome

Un atome consiste en un ou plusieurs caractères non spéciaux.

4.2 Nombre

Un nombre consiste en un ou plusieurs caractères de chiffres, et représente une valeur numérique.

4.3 Chaîne

Une chaîne est d'une des deux formes littérale ou guillemettée. La forme littérale est la forme générale de chaîne. La chaîne guillemettée est une solution de remplacement qui évite la redondance du traitement d'une chaîne littérale au prix des limitations de caractères qui peuvent être utilisés.

Une chaîne littérale est une séquence de zéro, un ou plusieurs octets (y compris CR et LF) préfixée d'un compte d'octets sous la forme d'une accolade ouverte ("{"), le nombre d'octets, une accolade fermée ("}"), et CRLF. Dans le cas de chaînes littérales transmises du serveur au client, le CRLF est immédiatement suivi par les octets de données. Dans le cas de chaînes littérales transmises du client au serveur, le client DOIT attendre de recevoir une demande de continuation de commande (décrite plus loin dans ce document) avant d'envoyer les octets de données (et le reste de la commande).

Une chaîne guillemettée est une séquence de zéro un ou plusieurs caractères de 7 bits, à l'exclusion de CR et LF, avec des guillemets (<">) à chaque extrémité.

La chaîne vide est représentée par "" (une chaîne guillemettée avec zéro caractère entre des doubles guillemets) ou par {} suivi par CRLF (une chaîne littérale avec un compte d'octet de 0).

Note : Même si le compte d'octet est 0, un client qui transmet une chaîne littérale DOIT attendre de recevoir une demande de continuation de commande.

4.3.1 Chaînes de 8 bits et chaînes binaires

La messagerie de texte à 8 bits et binaire est prise en charge par l'utilisation du codage de transfert de contenu MIME-IMB [RFC2045]. Les mises en œuvre de IMAP4rev1 PEUVENT transmettre des caractères de 8 bits ou multi-octet en littéral, mais DEVRAIENT ne faire ainsi que lorsque le CHARSET est identifié.

Bien qu'un codage de corps BINARY soit défini, les chaînes binaires non codées ne sont pas permises. Une "chaîne binaire" est toute chaîne avec des caractères NUL. Les mises en œuvre DOIVENT coder les données binaires dans une forme textuelle, telle que le BASE64, avant de transmettre les données. Une chaîne avec une quantité excessive de caractères CTL PEUT aussi être considérée comme étant binaire.

4.4 Liste entre parenthèses

Les structures de données sont représentées comme une "liste entre parenthèses"; une séquence d'éléments de données, délimités par une espace, et bordée à chaque extrémité par une parenthèse. Une liste entre parenthèses peut contenir d'autres

listes entre parenthèses, en utilisant plusieurs niveaux de parenthèses pour indiquer l'incorporation.

La liste vide est représentée par () – une liste entre parenthèses sans membre.

4.5 NIL

La forme spéciale "NIL" représente la non-existence d'un élément de données particulier qui est représenté par une chaîne ou liste entre parenthèses, pour la distinguer de la chaîne vide "" ou de la liste entre parenthèses vide ().

Note : NIL n'est jamais utilisé pour un élément de données qui prend la forme d'un atome. Par exemple, un nom de boîte aux lettres de "NIL" est un nom de boîte aux lettres NIL, par opposition à un nom de boîte aux lettres non existant. Ceci parce que les boîtes aux lettres utilisent une syntaxe "astring" qui est un atome ou une chaîne. À l'inverse, un addr-name de NIL est un nom personnel non existant, parce que addr-name utilise la syntaxe "nstring" qui est NIL ou une chaîne, mais jamais un atome.

5. Considérations pour le fonctionnement

Les règles suivantes sont énoncées ici pour assurer que toutes les mises en œuvre IMAP4rev1 interopèrent correctement.

5.1 Dénomination de boîte aux lettres

Les noms des boîtes aux lettres sont de 7 bits. Les mises en œuvre de client NE DOIVENT PAS tenter de créer de noms de boîte aux lettres de 8 bits, et DEVRAIENT interpréter tous les noms de boîte aux lettres de 8 bits retournés par LIST ou LSUB comme de l'UTF-8. Les mises en œuvre de serveur DEVRAIENT interdire la création de noms de boîte aux lettres de 8 bits, et NE DEVRAIENT PAS retourner de noms de boîte aux lettres de 8 bits dans LIST ou LSUB. Voir au paragraphe 5.1.3 des informations complémentaires sur la façon de représenter des noms de boîte aux lettres non-ASCII.

Note : les noms de boîte aux lettres de 8 bits étaient indéfinis dans les précédentes versions de ce protocole. Certains sites utilisaient un jeu de caractères local à 8 bits pour représenter des noms de boîte aux lettres non-ASCII. Un tel usage n'est pas interopérable, et est maintenant formellement déconseillé.

Le nom de boîte aux lettres insensible à la casse INBOX est un nom particulier réservé pour signifier "la principale boîte aux lettres pour cet usager sur ce serveur". L'interprétation de tous les autres noms dépend de la mise en œuvre.

En particulier, la présente spécification ne prend pas position sur la sensibilité à la casse dans les noms de boîte aux lettres non INBOX. Certaines mises en œuvre de serveur sont pleinement sensibles à la casse ; d'autres préservent la casse d'un nom nouvellement créé mais sont par ailleurs insensibles à la casse ; et d'autres encore obligent les noms à une casse particulière. Les mises en œuvre de client DOIVENT interagir avec toutes celles-ci. Si une mise en œuvre de serveur interprète des noms de boîte aux lettres non-INBOX comme insensibles à la casse, elle DOIT traiter les noms qui utilisent la convention internationale de dénomination précisément comme décrit au paragraphe 5.1.3.

Lors de la création d'un nouveau nom de boîte aux lettres, certaines considérations doivent être prises en compte à l'égard du client :

- 1) Tout caractère qui est un des atomes spéciaux (voir la syntaxe formelle) exigera que le nom de boîte aux lettres soit représenté par une chaîne guillemetée ou littérale.
- 2) Les CTL et autres caractères non graphiques sont difficiles à représenter dans une interface d'utilisateur et il vaudrait mieux les éviter.
- 3) Bien que les caractères génériques de liste ("% " et "*") soient valides dans un nom de boîte aux lettres, il est difficile d'utiliser de tels noms de boîte aux lettres avec les commandes LIST et LSUB du fait de conflits avec l'interprétation du caractère générique.
- 4) Habituellement, un caractère (déterminé par la mise en œuvre de serveur) est réservé pour délimiter les niveaux de hiérarchie.
- 5) Deux caractères, "#" et "&", ont une signification par convention, et devraient être évités sauf lorsque utilisés selon cette convention.

5.1.1 Hiérarchie de dénomination des boîtes aux lettres

Si on désire exporter des noms de boîte aux lettres hiérarchiques, les noms de boîte aux lettres DOIVENT être hiérarchisés de gauche à droite en utilisant un seul caractère pour séparer les niveaux hiérarchiques. Le même caractère séparateur de hiérarchie est utilisé pour tous les niveaux hiérarchiques au sein d'un seul nom.

5.1.2 Convention de dénomination de l'espace des noms de boîtes aux lettres

Par convention, le premier élément hiérarchique de tout nom de boîte aux lettres qui commence par "#" identifie "l'espace de nom" du reste du nom. Cela rend possible de faire la distinction entre différents types de mémorisations de boîtes aux lettres, chacun d'eux ayant son propre espace de nom.

Par exemple, les mises en œuvre qui offrent l'accès aux groupes de nouvelles USENET PEUVENT utiliser l'espace de nom "#news" pour séparer l'espace de nom de groupes de nouvelles USENET de celui des autres boîtes aux lettres. Et donc, le groupe de nouvelles comp.mail.misc aurait un nom de boîte aux lettres de "#news.comp.mail.misc", et le nom "comp.mail.misc" peut se référer à un objet différent (par exemple, la boîte aux lettres privée de l'utilisateur).

5.1.3 Convention internationale de dénomination des boîtes aux lettres

Par convention, les noms de boîtes aux lettres internationaux dans IMAP4rev1 sont spécifiés en utilisant une version modifiée du codage UTF-7 décrite dans [UTF-7]. L'UTF-7 modifié peut aussi être utilisable dans les serveurs qui mettent en œuvre une version antérieure du présent protocole.

En UTF-7 modifié, les caractères US-ASCII imprimables, excepté "&", représentent eux-mêmes, c'est-à-dire les caractères avec les valeurs d'octet 0x20 à 0x25 et 0x27 à 0x7e. Le caractère "&" (0x26) est représenté par la séquence de deux octets "&-".

Tous les autres caractères (valeurs d'octet 0x00 à 0x1f et 0x7f à 0xff) sont représentés en BASE64 modifié, avec une modification supplémentaire tirée de [UTF-7] qui est que "," est utilisé à la place de "/". Le BASE64 modifié NE DOIT PAS être utilisé pour représenter de caractères US-ASCII imprimables qui puissent se représenter eux-mêmes. Seuls les caractères de l'alphabet BASE64 modifié sont permis en texte BASE64 modifié. "&" est utilisé pour passer au BASE64 modifié et "-" pour revenir à l'US-ASCII. Il n'y a pas de glissement implicite du BASE64 à l'US-ASCII, et aucun glissement n'est permis ("-&" en BASE64 ; noter que "&-" en US-ASCII signifie "&"). Cependant, tous les noms débutent en US-ASCII, et DOIVENT se terminer en US-ASCII ; c'est-à-dire qu'un nom qui se termine par un caractère non ASCII ISO-10646 DOIT se terminer par un "-").

L'objet de ces modifications est de corriger les problèmes suivants de l'UTF-7 :

- 1) UTF-7 utilise le caractère "+" pour le glissement ; cela entre en conflit avec l'utilisation courante de "+" dans les noms de boîte aux lettres, en particulier les noms de groupe de nouvelles USENET.
- 2) Le codage de l'UTF-7 est le BASE64 qui utilise le caractère "/" ; cela entre en conflit avec l'utilisation de "/" comme délimiteur hiérarchique courant.
- 3) L'UTF-7 interdit l'usage non codé de "\" ; cela entre en conflit avec l'utilisation de "\" comme délimiteur hiérarchique courant.
- 4) L'UTF-7 interdit l'usage non codé de "~" ; cela entre en conflit avec l'utilisation de "~" dans certains serveurs comme indicateur de répertoire de rattachement.
- 5) L'UTF-7 permet plusieurs formes de remplacement pour représenter la même chaîne ; en particulier, les caractères US-ASCII imprimables peuvent être représentés sous forme codée.

Bien que l'UTF-7 modifié soit une convention, il établit certaines exigences sur le serveur qui traite un nom de boîte aux lettres avec un caractère "&" incorporé. En particulier, les mises en œuvre de serveur DOIVENT préserver la forme exacte de la portion de BASE64 modifié d'un nom en UTF-7 modifié et traiter ce texte comme sensible à la casse, même si les noms sont par ailleurs insensibles à la casse ou à casse normalisée.

Les mises en œuvre de serveur DEVRAIENT vérifier que tout nom de boîte aux lettres avec un caractère "&" incorporé, utilisé comme argument de CREATE, est : en syntaxe UTF-7 correctement modifiée, n'a pas de glissements superflus, et n'a pas de codage en BASE64 modifié de caractère US-ASCII imprimable qui puisse se représenter lui-même. Cependant, les mises en œuvre de client NE DOIVENT PAS dépendre du serveur pour faire cela, et NE DEVRAIENT PAS tenter de créer un nom de boîte aux lettres avec un caractère "&" incorporé à moins qu'il ne soit conforme à la syntaxe UTF-7 modifiée.

Les mises en œuvre de serveur qui exportent un message mémorisé qui ne suit pas la convention UTF-7 modifiée DOIVENT convertir en UTF-7 modifié tout nom de boîte aux lettres qui contient des caractères non ASCII ou le caractère "&".

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

Par exemple, la chaîne "&Jjo!" n'est pas un nom de boîte aux lettres valide parce qu'elle ne contient pas de glissement à l'US-ASCII avant le "!". La forme correcte est "&Jjo-!". La chaîne "&U,BTFw-/&ZeVnLIqe-" n'est pas permise parce qu'elle contient un glissement superflu. La forme correcte est "&U,BTF2XIZyyKng-".

5.2 Taille de boîte aux lettres et mises à jour d'état de message

À tout moment, un serveur peut envoyer des données que le client n'a pas demandées. Parfois, un tel comportement est EXIGÉ. Par exemple, les agents autres que le serveur PEUVENT ajouter des messages à la boîte aux lettres (par exemple, nouvelle livraison de message), changer les fanions des messages dans la boîte aux lettres (par exemple, accès simultané à la même boîte aux lettres par plusieurs agents), ou même retirer des messages de la boîte aux lettres. Un serveur DOIT envoyer automatiquement des mises à jour de taille de boîte aux lettres si un changement de taille de boîte aux lettres est observé durant le traitement d'une commande. Un serveur DEVRAIT envoyer automatiquement des mises à jour de fanion de message, sans exiger du client qu'il demande explicitement de telles mises à jour.

Des règles particulières existent pour la notification d'un serveur à un client de l'effacement de messages pour prévenir des erreurs de synchronisation ; voir la description de la réponse EXPUNGE pour des précisions. En particulier, il N'EST PAS permis d'envoyer une réponse EXISTS qui réduirait le nombre des messages dans la boîte aux lettres ; seule la réponse EXPUNGE peut le faire.

Sans considération des décisions de mise en œuvre que prend un client pour récupérer des données auprès du serveur, une mise en œuvre de client DOIT enregistrer les mises à jour de taille de boîte aux lettres. Il NE DOIT PAS supposer qu'une commande émise après le choix initial de boîte aux lettres va retourner la taille de boîte aux lettres.

5.3 Réponse quand aucune commande n'est en cours

Il est permis aux mises en œuvre de serveur d'envoyer une réponse non étiquetée (sauf pour EXPUNGE) alors qu'il n'y a pas de commande en cours. Les mises en œuvre de serveur qui envoient de telles réponses DOIVENT prendre en compte les considérations de contrôle de flux. En particulier, elles DOIVENT soit (1) vérifier que la taille des données n'excède pas la taille de fenêtre disponible du transport sous-jacent, soit (2) utiliser des écritures non bloquantes.

5.4 Temporisateur d'auto déconnexion

Si un serveur a un temporisateur d'auto déconnexion en cas d'inactivité qui s'applique aux sessions après authentification, la durée de ce temporisateur DOIT être d'au moins 30 minutes. La réception de TOUTE commande durant cet intervalle DEVRAIT suffire à remettre à zéro le temporisateur d'auto déconnexion.

5.5 Plusieurs commandes en cours

Le client PEUT envoyer une autre commande sans attendre la réponse de résultat d'achèvement d'une commande, sous réserve des règles d'ambiguïté (voir ci-dessous) et des contraintes de contrôle de flux du flux de données sous-jacent. De même, un serveur PEUT commencer à traiter une autre commande avant de traiter la commande en cours et la mener à son terme, sous réserve des règles d'ambiguïté. Cependant, toute les réponses aux demandes de continuation de commande et les continuations de commandes DOIVENT être négociées avant l'initiation de toute commande suivante.

L'exception est si une ambiguïté devait résulter de ce qu'une commande affecterait les résultats d'autres commandes. Les clients NE DOIVENT PAS envoyer plusieurs commandes sans attendre si une ambiguïté pourrait en résulter. Si le serveur détecte une possible ambiguïté, il DOIT exécuter l'achèvement des commandes dans l'ordre donné par le client.

Le plus évident exemple d'ambiguïté est celui où une commande affecterait les résultats d'une autre commande, par exemple, un FETCH (*aller chercher*) des fanions d'un message et un STORE (*mémoriser*) des fanions de ce même message.

Une ambiguïté non évidente survient avec des commandes qui permettent une réponse EXPUNGE non étiquetée (des commandes autres que FETCH, STORE, et SEARCH), car une réponse EXPUNGE non étiquetée peut invalider les numéros de séquence dans une commande suivante. Cela ne pose pas de problème avec les commandes FETCH, STORE, ou SEARCH car il est interdit aux serveurs d'envoyer des réponses EXPUNGE lorsque l'une de ces commandes est en cours. Donc, si le client envoie toute commande autre que FETCH, STORE, ou SEARCH, il DOIT attendre la réponse de résultat d'achèvement avant d'envoyer une commande avec le numéro de séquence du message.

Note : les réponses EXPUNGE sont permises alors que UID FETCH, UID STORE, et UID SEARCH sont en cours. Si le client envoie une commande UID, il doit attendre une réponse de résultat d'achèvement avant d'envoyer une commande qui utilise le numéro de séquence du message (cela peut inclure UID SEARCH). Tout numéro de séquence du message dans un argument de UID SEARCH est associé au message avant l'effet de tout EXPUNGE non étiqueté retourné par UID SEARCH.

Par exemple, les séquences de commandes sans attente suivantes sont invalides :

```
FETCH + NOOP + STORE
STORE + COPY + FETCH
COPY + COPY
CHECK + FETCH
```

Voici des exemples de séquences de commandes sans attente valides :

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

UID SEARCH + UID SEARCH peut être valide ou invalide comme séquence de commande sans attente, selon que le second UID SEARCH contient ou non le numéro de séquence du message.

6. Commandes du client

Les commandes IMAP4rev1 sont décrites dans la présente section. Les commandes sont organisées par l'état dans lequel la commande est permise. Les commandes qui sont permises dans plusieurs états figurent dans l'état minimum permis (par exemple, les commandes valides dans l'état Authentifié et Choisi figurent dans les commandes d'état Authentifié).

Les arguments des commandes, identifiés par "Arguments:" dans les descriptions de commande ci-dessous, sont décrits par fonction, et non par la syntaxe. La syntaxe précise des arguments de commande est décrite dans la section 9 sur la syntaxe formelle.

Certaines commandes causent des réponses spécifiques à retourner par le serveur ; elles sont identifiées par "Réponses:" dans la descriptions de commande ci-dessous. Voir les descriptions des réponses dans la section Réponses pour des informations sur ces réponses, et la section Syntaxe formelle pour la syntaxe précise de ces réponses. Il est possible que les données du serveur soient transmises par suite de toute commande. Et donc, les commandes qui n'exigent pas spécifiquement de données du serveur spécifient "pas de réponses spécifiques pour cette commande" au lieu de "aucun".

Le "Résultat:" dans la description de commande se réfère aux possibles réponses d'état étiquetées à une commande, et à toute interprétation particulière de ces réponses d'état.

L'état d'une connexion n'est changé que par des commandes réussies qui sont décrites comme changeant l'état. Une commande rejetée (réponse BAD) ne change jamais l'état de la connexion ou de la boîte aux lettres choisie. Un échec de commande (réponse NO) ne change généralement pas l'état de la connexion ou de la boîte aux lettres choisie ; l'exception étant les commandes SELECT et EXAMINE.

6.1 Commandes du client – tous états

Les commandes suivantes sont valides dans tous les états : CAPABILITY, NOOP, et LOGOUT.

6.1.1 Commande CAPABILITY

Arguments : aucun

Réponses : EXIGE la réponse non étiquetée : CAPABILITY

Résultat : OK – capacité réalisée
BAD – commande inconnue ou argument invalide

La commande CAPABILITY demande une liste des capacités prises en charge par ce serveur. Le serveur DOIT envoyer une seule réponse CAPABILITY non étiquetée avec "IMAP4rev1" parmi les capacités de la liste avant la réponse OK (étiquetée).

Un nom de capacité qui commence par "AUTH=" indique que le serveur prend en charge ce mécanisme d'authentification particulier. Tous les noms de cette sorte font, par définition, partie de la présente spécification. Par exemple, la capacité d'autorisation pour une authentification expérimentale "blurdybloop" serait "AUTH=XBLURDYBLOOP" et non "XAUTH=BLURDYBLOOP" ou "XAUTH=XBLURDYBLOOP".

Les autres noms de capacité se réfèrent aux extensions, révisions, ou amendements à cette spécification. Voir la documentation sur la réponse CAPABILITY pour des informations complémentaires. Aucune capacité, au delà de l'ensemble IMAP4rev1 de base défini dans la présente spécification, n'est activée sans une action explicite du client pour invoquer cette capacité.

Les mises en œuvre de serveurs et de clients DOIVENT mettre en œuvre les capacités STARTTLS, LOGINDISABLED et AUTH=PLAIN (décrites dans la [RFC2595]). Des informations importantes figurent dans la Section 11 "Considérations pour le sécurité".

Voir au paragraphe intitulé "Commandes du client - Expérimental/Expansion" pour des informations sur la forme de capacités spécifiques d'un site ou d'une mise en œuvre .

Exemple :

```
C: abcd CAPABILITY
S: * CAPABILITY IMAP4rev1 STARTTLS AUTH=GSSAPI LOGINDISABLED
S: abcd OK CAPABILITY terminé
C: efgh STARTTLS
S: efgh OK STARTTLS terminé <négociation TLS , les autres commandes sont sous la couche [TLS]>
C: ijkl CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=GSSAPI AUTH=PLAIN
S: ijkl OK CAPABILITY terminé
```

6.1.2 Commande NOOP

Argument : aucun

Réponses : pas de réponses spécifiques pour cette commande (mais voir ci-dessous)

Résultat : OK - noop terminé
BAD – commande inconnue ou arguments invalide

La commande NOOP réussit toujours. Elle ne fait rien.

Comme toute commande peut retourner une mise à jour d'état sous forme de données non étiquetées, la commande NOOP peut être utilisée comme interrogation périodique des nouveaux messages ou comme mise à jour de l'état des messages durant une période d'inactivité (c'est la méthode préférée pour ce faire). La commande NOOP peut aussi être utilisée pour rétablir un temporisateur d'auto déconnexion sur inactivité au serveur.

Exemple :

```
C: a002 NOOP
S: a002 OK NOOP terminé
...
C: a047 NOOP
S: * 22 EXPUNGE
S: * 23 EXISTS
S: * 3 RECENT
S: * 14 FETCH (FLAGS (\vu \supprimé))
```

S: a047 OK NOOP terminé

6.1.3 Commande LOGOUT

Argument : aucun

Réponses : EXIGE une réponse : BYE non étiquetée

Résultat : OK – déconnexion achevée
BAD – commande inconnue ou arguments invalides

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

Exemple :

C: A023 LOGOUT

S: * BYE IMAP4rev1 Déconnexion du serveur

S: A023 OK LOGOUT achevée

(Serveur et client closent alors la connexion)

6.2 Commandes du client – état Non authentifié

Dans l'état Non authentifié, la commande AUTHENTICATE ou LOGIN établit l'authentification et entre dans l'état Authentifié. La commande AUTHENTICATE fournit un mécanisme général pour diverses techniques d'authentification, de protection de la confidentialité, et de vérification d'intégrité, car la commande LOGIN utilise une paire traditionnelle de nom d'utilisateur et mot de passe en clair et n'a pas de moyen d'établir de protection de la confidentialité ou de vérification d'intégrité.

La commande STARTTLS est une autre forme d'établissement de la protection de la confidentialité de la session et de vérification d'intégrité, mais n'établit pas l'authentification ou n'entre pas dans l'état authentifié.

Les mises en œuvre de serveur PEUVENT permettre l'accès à certaines boîtes aux lettres sans établir l'authentification. Cela peut être fait au moyen de l'authentifiant ANONYMOUS [SASL] décrit dans la [RFC 2245]. Une convention plus ancienne est la commande LOGIN qui utilise l'identifiant d'utilisateur (*userid*) "anonymous" ; dans ce cas, un mot de passe est exigé bien que le serveur puisse choisir d'accepter tout mot de passe. Les restrictions opposées aux utilisateurs anonymes dépendent de la mise en œuvre.

Une fois authentifié (y compris comme anonyme), il n'est plus possible de revenir à l'état Non authentifié.

En plus des commandes universelles (CAPABILITY, NOOP, et LOGOUT), les commandes suivantes sont valides dans l'état Non authentifié : STARTTLS, AUTHENTICATE et LOGIN. Voir à la Section 11 "Considérations sur la sécurité" des informations importantes sur ces commandes.

6.2.1 Commande STARTTLS

Argument : aucun

Réponses : pas de réponse spécifique pour cette commande.

Résultat : OK - starttls achevé, début de négociation TLS.
BAD - commande inconnue ou arguments invalides

Une négociation TLS [RFC2246] commence immédiatement après le CRLF à la fin de la réponse OK étiquetée du serveur. Une fois qu'un client a produit une commande STARTTLS, il NE DOIT PAS produire d'autres commandes jusqu'à ce qu'une réponse de serveur soit vue et que la négociation TLS soit achevée.

Le serveur reste dans l'état Non authentifié, même si les accreditifs du client sont fournis durant la négociation TLS. Cela n'empêche pas un mécanisme d'authentification tel que EXTERNAL (défini dans [SASL]) d'utiliser l'identité du client déterminée par la négociation TLS.

Une fois que TLS a été commencé, le client DOIT éliminer les informations mises en antémémoire sur les capacités du

serveur et DEVRAIT réémettre la commande CAPABILITY. Ceci est nécessaire pour protéger contre les attaques par interposition (*man-in-the-middle attack*) qui altèrent la liste des capacités avant STARTTLS. Le serveur PEUT annoncer des capacités différentes, et en particulier NE DEVRAIT PAS annoncer de capacité STARTTLS, après une commande STARTTLS réussie.

Exemple :

```
C: a001 CAPABILITY
S: * CAPABILITY IMAP4rev1 STARTTLS LOGINDISABLED
S: a001 OK CAPABILITY achevée
C: a002 STARTTLS
S: a002 OK Commencer maintenant la négociation TLS
                                     <négociation TLS, les commandes suivantes sont sous la couche TLS>
C: a003 CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=PLAIN
S: a003 OK CAPABILITY achevée
C: a004 LOGIN mot de passe joe
S: a004 OK LOGIN achevé
```

6.2.2 Commande AUTHENTICATE

Argument : nom du mécanisme d'authentification

Réponses : des données de continuation peuvent être nécessaires

Résultat : OK – authentification achevée, on est maintenant dans l'état Authentifié
 NO – échec d'authentification : mécanisme d'authentification non accepté, accreditifs rejetés
 BAD - commande inconnue ou arguments invalides, échange d'authentification annulé

La commande AUTHENTICATE indique un mécanisme d'authentification SASL au serveur. Si le serveur prend en charge le mécanisme d'authentification demandé, il effectue un échange du protocole d'authentification pour authentifier et identifier le client. Il PEUT aussi négocier une couche de sécurité FACULTATIVE pour des interactions de protocole ultérieures. Si le mécanisme d'authentification demandé n'est pas pris en charge, le serveur DEVRAIT rejeter la commande AUTHENTICATE en envoyant une réponse NO étiquetée.

La commande AUTHENTICATE ne prend pas en charge la caractéristique facultative "réponse initiale" de SASL. Le paragraphe 5.1 de la [RFC2222] spécifie comment traiter un mécanisme d'authentification qui utilise une réponse initiale.

Le nom de service spécifié par ce profil de protocole de SASL est "imap".

L'échange du protocole d'authentification consiste en une série de mises en causes du serveur et de réponses du client qui sont spécifiques du mécanisme d'authentification. Une mise en cause de serveur consiste en une réponse de demande de continuation de commande avec le jeton "+" suivi d'une chaîne codée en BASE64. La réponse du client comporte une seule ligne consistant en une chaîne codée en BASE64. Si le client souhaite annuler un échange d'authentification, il produit une ligne consistant en un seul "*". Si le serveur reçoit une telle réponse, ou si il reçoit une chaîne invalide en BASE64 (par exemple des caractères étrangers à l'alphabet BASE64, ou un "=" non terminal), il DOIT rejeter la commande AUTHENTICATE en envoyant une réponse BAD étiquetée.

Si une couche de sécurité est négociée au moyen de l'échange d'authentification SASL, elle prend effet immédiatement après le CRLF qui conclut l'échange d'authentification pour le client, et le CRLF de la réponse OK étiquetée pour le serveur.

Alors que les mises en œuvre de clients et de serveurs DOIVENT mettre en œuvre la commande AUTHENTICATE elle-même, il n'est pas exigé de mettre en œuvre de mécanismes d'authentification autres que le mécanisme PLAIN décrit dans la [RFC2595]. Aussi, il n'est pas exigé qu'un mécanisme d'authentification prenne en charge une couche de sécurité.

Note : une mise en œuvre de serveur DOIT mettre en œuvre une configuration dans laquelle elle ne permet aucun mécanisme de mot de passe en clair, sauf si la commande STARTTLS a été négociée ou si quelque autre mécanisme qui protège la session contre la divulgation du mot de passe a été fourni. Les sites de serveurs NE DEVRAIENT PAS utiliser de configuration qui permette un mécanisme de mot de passe en clair sans un tel mécanisme de protection contre la divulgation de mot de passe. Les mises en œuvre de clients et de serveurs DEVRAIENT mettre en œuvre des mécanismes SASL supplémentaires qui n'utilisent pas de mots de passe en clair, tels que le mécanisme GSSAPI décrit dans la [RFC2222] et/ou le mécanisme de résumé MD-5 de la [RFC2831].

Les serveurs et les clients peuvent prendre en charge plusieurs mécanismes d'authentification. Le serveur DEVRAIT faire une liste des mécanismes d'authentification qu'il prend en charge dans la réponse à la commande CAPABILITY de façon que le client sache quels mécanismes d'authentification utiliser.

Un serveur PEUT inclure un code de réponse CAPABILITY dans la réponse OK étiquetée d'une commande AUTHENTICATE réussie afin d'envoyer automatiquement ses capacités. Il n'est pas nécessaire pour un client d'envoyer une commande CAPABILITY séparée si il reconnaît ces capacités automatiques. Cela ne devrait être fait que si il n'a pas été négocié de couche de sécurité par la commande AUTHENTICATE, parce que la réponse OK étiquetée au titre d'une commande AUTHENTICATE n'est pas protégée par la vérification de chiffrement/intégrité. SASL [RFC2222] exige du client qu'il réémette une commande CAPABILITY dans ce cas.

Si une commande AUTHENTICATE échoue avec une réponse NO, le client PEUT essayer un autre mécanisme d'authentification en produisant une autre commande AUTHENTICATE. Il PEUT aussi tenter de s'authentifier en utilisant la commande LOGIN (voir au paragraphe 6.2.3 des détails complémentaires). En d'autres termes, le client PEUT demander des types d'authentification en ordre de préférences décroissantes, avec la commande LOGIN en dernier ressort.

L'identité d'autorisation passée du client au serveur durant l'échange d'authentification est interprétée par le serveur comme le nom d'utilisateur dont le client demande les privilèges.

Exemple :

```
S: * OK IMAP4rev1 Server
C: A001 AUTHENTICATE GSSAPI
S: +
C:
YIIB+wYJKoZIhvcSAQICAQBuggHqMIIB5qADAgEFoQMCAQ6iBwMFACAAAACjggEmYYIBIjCCAR6gAwIBBaE
SGxB1Lndhc2hpbmd0b24uZWR1oi0wK6ADAgEDoSQwIhsEaW1hcBbac2hpdmFtcy5jYWwud2FzaGluZ3Rvbi5lZHWjg
dMwgdCgAwIBAAEDAgEDooHDBIHAcS1GSa5b+fXnPZNMxB9SjL8Ollj2SKyb+3S0iXmIjen/jNkpJXAleKTz6BQPzj
8duz8EtoOuNfKgweViyn/9B9bccy1uuAE2HI0yC/PHXNNU9ZrBziJ8Lm0tTnc98kUpjXnHZhsMcz5Mx2GR6dGknbl0ia
GcRerMUsWOUbMkkRmVMMdR9T3EZdpqsBd7jZCNMWotjhivd5zovQIFqQ2Wjc2+y46vKP/iXxWtuQJuDiisyXF0
Y8+5GTpALpHDcl/pIGmMIGjoAMCAQGigZsEgZg2on5mSuxoDHEA1w9bcW9nFdfXDKpdrQhVGVRDIzcCMCTzv
Uboqb5KjY1NJKJsfjRQiBYBdENKfzK+g5DIV8nrw81uOcP8NOQCLR5XkoMHC0Dr/80ziQzbNqhxO6652Npft0LQwJv
enwDI13YxpwOdMXzkWZN/XrEqOWp6GCgXTBvCyLWLIWnbaUkZdEYbKHBPjd8t/1x5Yg==
S: +
YGgGCSqGSIb3EgECAgIAb1kwV6ADAgEFoQMCAQ+iSzBjoAMCAQGiQgRAtHTEuOP2BXb9sBYFR4SJIDZxmg3
9IxmRBOhXRkdDA0uHTCOT9Bq3OsUTXUlK0CsFLoa8j+gvGDlgHuqzWHPSQg==
C:
S: + YDMGCSqGSIb3EgECAgIBAAD/////6jcyG4GE3KkTzBeBiVHeceP2CWY0SR0fAQAgAAQEBAQ=
C: YDMGCSqGSIb3EgECAgIBAAD/////3LQBHXTPfFZgrejpLILImPwkhbfa2QteAQAgAG1yYwE=
S: A001 OK GSSAPI authentification réussie
```

Note : Les sauts de ligne au sein de la mise en cause du serveur et des réponses du client sont pour la facilité de lecture et ne sont pas dans les authentifiants réels.

6.2.3 Commande LOGIN

Arguments : mot de passe de nom d'usager

Réponses : pas de réponses spécifiques pour cette commande

Résultat : OK – connexion achevée, maintenant dans l'état Authentifié
 NO – échec de connexion : nom d'utilisateur ou mot de passe rejeté
 BAD – commande inconnue ou arguments invalides

La commande LOGIN identifie le client auprès du serveur et porte le mot de passe en clair qui authentifie cet usager.

Un serveur PEUT inclure un code de réponse CAPABILITY dans la réponse OK étiquetée à une commande LOGIN réussie afin d'envoyer automatiquement les capacités. Il n'est pas nécessaire qu'un client envoie une commande CAPABILITY séparée si il reconnaît ces capacités automatiques.

Exemple :

```
C: a001 LOGIN SMITH SESAME
```

S: a001 OK LOGIN achevé

Note : L'utilisation de la commande LOGIN sur un réseau non sûr (comme l'Internet) présente un risque pour la sécurité, parce que quiconque surveille le trafic du réseau peut obtenir les mots de passe en clair. La commande LOGIN NE DEVRAIT PAS être utilisée sauf en dernier ressort, et il est recommandé que les mises en œuvre de client aient un moyen de désactiver toute utilisation automatique de la commande LOGIN.

Sauf si la commande STARTTLS a été négociée ou si est fourni quelque autre mécanisme protégeant la session contre la divulgation du mot de passe, une mise en œuvre de serveur DOIT mettre en œuvre une configuration dans laquelle elle annonce la capacité LOGINDISABLED et NE DOIT PAS permettre la commande LOGIN. Les sites de serveur NE DEVRAIENT PAS utiliser de configuration qui permette la commande LOGIN sans un tel mécanisme de protection contre la divulgation de mot de passe. Une mise en œuvre de client NE DOIT PAS envoyer une commande LOGIN si la capacité LOGINDISABLED est annoncée.

6.3 Commandes du client – état Authentifié

Dans l'état Authentifié, les commandes qui manipulent les boîtes aux lettres comme des entités atomiques sont permises. Parmi ces commandes, SELECT et EXAMINE vont sélectionner une boîte aux lettres en accès et entrer dans l'état Choisi.

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

6.3.1 Commande SELECT

Argument : nom de boîte aux lettres

Réponses : réponses EXIGÉES non étiquetée : FLAGS, EXISTS, RECENT
réponses EXIGÉES OK non étiquetées : UNSEEN, PERMANENTFLAGS, UIDNEXT, UIDVALIDITY

Résultat : OK – sélection achevée, maintenant dans l'état Choisi
NO – échec de sélection, maintenant dans l'état Authentifié : pas de telle boîte aux lettres, pas d'accès à la boîte aux lettres
BAD – commande inconnue ou arguments invalides

La commande SELECT choisit une boîte aux lettres de telle sorte qu'on puisse accéder aux messages de la boîte aux lettres. Avant de retourner un OK au client, le serveur DOIT envoyer les données non étiquetées suivantes au client. Noter que les versions précédentes de ce protocole n'exigeaient que les données FLAGS, EXISTS, et RECENT non étiquetées ; par conséquent, les mises en œuvre de client DEVRAIENT mettre en œuvre le comportement par défaut pour les données manquantes comme exposé dans les éléments individuels.

FLAGS Les fanions définis dans la boîte aux lettres. Voir la description de la réponse à FLAGS pour les détails.

<n> EXISTS Le nombre de messages dans la boîte aux lettres. Voir la description de la réponse EXISTS pour les détails.

<n> RECENT Le nombre de messages avec le fanion \Récent mis. Voir la description de la réponse RECENT pour les détails.

OK [UNSEEN <n>]

Le numéro de séquence du message du premier message non vu dans la boîte aux lettres. Si il manque, le client ne peut faire aucune hypothèse sur le premier message non vu dans la boîte aux lettres, et doit produire une commande SEARCH si il veut le trouver.

OK [PERMANENTFLAGS (<liste des fanions>)]

C'est une liste des fanions de message que le client peut changer de façon permanente. Si il manque, le client devrait supposer que tous les fanions peuvent être changés de façon permanente.

OK [UIDNEXT <n>]

C'est la prochaine valeur d'identifiant univoque. Se reporter au paragraphe 2.3.1.1 pour plus d'informations. Si il manque, le client ne peut faire aucune hypothèse sur la prochaine valeur d'identifiant

univoque.

OK [UIDVALIDITY <n>]

C'est la valeur de la validité de l'identifiant univoque. Se reporter au paragraphe 2.3.1.1 pour plus d'informations. Si il manque, le serveur ne prend pas en charge les identifiants univoques.

Une seule boîte aux lettres peut être choisie à la fois dans une connexion ; les accès simultanés à plusieurs boîtes aux lettres exigent plusieurs connexions. La commande SELECT désélectionne automatiquement toute boîte aux lettres actuellement sélectionnée avant de tenter la nouvelle sélection. Par conséquent, si une boîte aux lettres est choisie et si est tentée une commande SELECT qui échoue, aucune boîte aux lettres n'est sélectionnée.

Si il est permis au client de modifier la boîte aux lettres, le serveur DEVRAIT préfixer le texte de la réponse OK étiquetée du code de réponse "[READ-WRITE]" (*lecture-écriture*).

Si il n'est pas permis au client de modifier la boîte aux lettres mais si un accès en lecture est permis, la boîte aux lettres est sélectionnée en lecture seule, et le serveur DOIT préfixer le texte de la réponse OK étiquetée à SELECT du code de réponse "[READ-ONLY]" (*lecture seule*). L'accès en lecture seule par SELECT diffère de la commande EXAMINE en ce que certaines boîtes aux lettres en lecture seule PEUVENT permettre de changer l'état permanent selon l'utilisateur (par opposition à une base globale). Les messages de groupes de nouvelles marqués dans un fichier .newsrsrc appuyé sur un serveur sont un exemple d'un tel état permanent selon l'utilisateur qui peuvent être modifiés dans une boîtes aux lettres en lecture seule.

Exemple :

```
C: A142 SELECT INBOX
S: * 172 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Le message 12 est le premier non vu
S: * OK [UIDVALIDITY 3857529045] Les UID sont valides
S: * OK [UIDNEXT 4392] Prochain UID prévu
S: * FLAGS (\Répondu\Fanion \Supprimé\Vu \Brouillon)
S: * OK [PERMANENTFLAGS (\Supprimé\Vu \*)] Limité
S: A142 OK [READ-WRITE] SELECT achevé
```

6.3.2 Commande EXAMINE

Argument : nom de boîte aux lettres

Réponses : réponses EXIGÉES non étiquetée : FLAGS, EXISTS, RECENT
réponses EXIGÉES OK non étiquetées : UNSEEN, PERMANENTFLAGS, UIDNEXT, UIDVALIDITY

Résultat : OK – examen achevé, maintenant dans l'état Choisi
NO – échec de l'examen, maintenant dans l'état Authentifié : pas de telle boîte aux lettres, pas d'accès à la boîte aux lettres
BAD – commande inconnue ou arguments invalides

La commande EXAMINE est identique à SELECT et retourne le même résultat ; cependant, la boîte aux lettres choisie est identifiée en lecture seule. Aucun changement à l'état permanent de la boîte aux lettres, y compris l'état selon l'utilisateur, n'est permis; en particulier, EXAMINE NE DOIT PAS être cause que les messages perdent le fanion \Récent.

Le texte des réponses OK étiquetées à la commande EXAMINE DOIT commencer par le code de réponse "[READ-ONLY]".

Exemple :

```
C: A932 EXAMINE blurdybloop
S: * 17 EXISTS
S: * 2 RECENT
S: * OK [UNSEEN 8] Le message 8 est le premier non vu
S: * OK [UIDVALIDITY 3857529045] Les UID sont valides
S: * OK [UIDNEXT 4392] Prochain UID prévu
S: * FLAGS (\Répondu\Fanion \Supprimé\Vu \Brouillon)
S: * OK [PERMANENTFLAGS ()] Pas de fanion permanent permis
S: A932 OK [READ-ONLY] EXAMINE achevé
```

6.3.3 Commande CREATE

Argument : nom de boîte aux lettres

Réponses : pas de réponses spécifiques pour cette commande

Résultat : OK – création achevée
NO – échec de création : on ne peut pas créer de boîte aux lettres avec ce nom
BAD – commande inconnue ou arguments invalides

La commande CREATE crée une boîte aux lettres avec le nom donné. Une réponse OK n'est retournée que si une nouvelle boîte aux lettres a été créée avec ce nom. C'est une erreur de tenter de créer INBOX ou une boîte aux lettres avec un nom qui se réfère à une boîte aux lettres existante. Toute erreur de création va retourner une réponse NO étiquetée.

Si le nom de boîte aux lettres est suffixé avec le caractère séparateur de hiérarchie du serveur (tel que retourné du serveur par une commande LIST) cela déclare que le client a l'intention de créer un nom de boîte aux lettres sous ce nom dans la hiérarchie. Les mises en œuvre de serveur qui n'exigent pas cette déclaration DOIVENT ignorer la déclaration. Dans tous les cas, le nom créé l'est sans délimiteur hiérarchique en queue.

Si le caractère séparateur hiérarchique du serveur apparaît ailleurs dans le nom, le serveur DEVRAIT créer tous les noms supérieurs dans la hiérarchie qui sont nécessaires pour que la commande CREATE s'achève avec succès. En d'autres termes, une tentative de création de "foo/bar/zap" sur un serveur dans lequel "/" est le caractère séparateur hiérarchique DEVRAIT créer foo/ et foo/bar/ si ils n'existent pas déjà.

Si une nouvelle boîte aux lettres est créée avec le même nom qu'une boîte aux lettres supprimée, son identifiant univoque DOIT être supérieur à tout identifiant univoque utilisé dans l'incarnation précédente de la boîte aux lettres SAUF si la nouvelle incarnation a une valeur d'identifiant univoque différente. Voir la description de la commande UID pour des précisions.

Exemple :

```
C: A003 CREATE owatagusiam/  
S: A003 OK CREATE achevé  
C: A004 CREATE owatagusiam/blurdybloop  
S: A004 OK CREATE achevé
```

Note : L'interprétation de cet exemple dépend de si "/" a été retourné comme séparateur hiérarchique à partir de LIST. Si "/" est le séparateur hiérarchique, un nouveau niveau hiérarchique dénommé "owatagusiam" avec un membre appelé "blurdybloop" est créé. Autrement, deux boîtes aux lettres sont créées au même niveau hiérarchique.

6.3.4 Commande DELETE

Argument : nom de boîte aux lettres

Réponses : pas de réponses spécifiques pour cette commande

Résultat : OK – suppression achevée
NO – échec de suppression : on ne peut supprimer la boîte aux lettres qui porte ce nom
BAD – commande inconnue ou arguments invalides

La commande DELETE supprime de façon permanente la boîte aux lettres du nom donné. Une réponse OK étiquetée n'est retournée que si la boîte aux lettres a été supprimée. C'est une erreur de tenter de supprimer INBOX ou un nom de boîte aux lettres qui n'existe pas.

La commande DELETE NE DOIT PAS supprimer les noms inférieurs de la hiérarchie. Par exemple, si une boîte aux lettres "foo" a un inférieur "foo.bar" (en supposant que "." est le caractère de délimitation hiérarchique) supprimer "foo" NE DOIT PAS supprimer "foo.bar". C'est une erreur de tenter de supprimer un nom qui a des noms hiérarchiquement inférieurs et aussi qui a l'attribut \Noselect de nom de boîte aux lettres (voir la description de la réponse LIST pour les détails).

Il est permis de supprimer un nom qui a des noms hiérarchiques inférieurs et qui n'a pas l'attribut \Noselect de nom de boîte aux lettres. Si la mise en œuvre de serveur ne permet pas de supprimer le nom alors qu'existent des noms hiérarchiques inférieurs, l'attribut \Noselect de nom de boîte aux lettres est mis pour ce nom. Dans tous les cas, tous les messages dans cette boîte aux lettres sont supprimés par la commande DELETE.

La valeur du plus fort identifiant univoque utilisé de la boîte aux lettres supprimée DOIT être préservée afin qu'une nouvelle boîte aux lettres créée avec le même nom ne réutilise pas les identifiants de la précédente incarnation, SAUF si la nouvelle incarnation a une valeur différente de validité d'identifiant univoque. Voir la description de la commande UID pour les détails.

Exemples :

```
C: A682 LIST "" *
S: * LIST () "/" blurdybloop
S: * LIST (\Noselect) "/" foo
S: * LIST () "/" foo/bar
S: A682 OK LIST achevé
C: A683 DELETE blurdybloop
S: A683 OK DELETE achevé
C: A684 DELETE foo
S: A684 NO Le nom "foo" a des noms hiérarchiques inférieurs
C: A685 DELETE foo/bar
S: A685 OK DELETE achevé
C: A686 LIST "" *
S: * LIST (\Noselect) "/" foo
S: A686 OK LIST achevé
C: A687 DELETE foo
S: A687 OK DELETE achevé
```

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

6.3.5 Commande RENAME

Arguments : nom de boîte aux lettres existant
nouveau nom de boîte aux lettres

Réponses : pas de réponses spécifiques pour cette commande

Résultat : OK – changement de dénomination achevé.
NO – échec de changement de dénomination : pas possible de désigner la boîte aux lettres par ce nom, pas possible de renommer en boîte aux lettres avec ce nom.
BAD – commande inconnue ou arguments invalides.

La commande RENAME change le nom d'une boîte aux lettres. Une réponse OK étiquetée n'est retournée que si la boîte aux lettres a été renommée. C'est une erreur de tenter de renommer à partir d'un nom de boîte aux lettres qui n'existe pas ou en un nom de boîte aux lettres qui existe déjà. Toute erreur de renommage retournera une réponse NO étiquetée.

Si le nom a des noms hiérarchiques inférieurs, ceux-ci DOIVENT aussi être renommés. Par exemple, le changement de nom de "foo" en "zap" va renommer "foo/bar" (en supposant que "/" est le caractère de délimitation hiérarchique) en "zap/bar".

Si le caractère de séparation hiérarchique du serveur apparaît dans le nom, le serveur DEVRAIT créer tous les noms hiérarchiques supérieurs qui sont nécessaires pour que la commande RENAME s'achève avec succès. En d'autres termes, une tentative de renommer "foo/bar/zap" en baz/rag/zowie sur un serveur dans lequel "/" est le caractère de séparation

hiérarchique DEVRAIT créer baz/ et baz/rag/ si ils n'existent pas déjà.

La valeur du plus fort identifiant univoque utilisé de l'ancien nom de boîte aux lettres DOIT être préservé afin qu'une nouvelle boîte aux lettres créée avec le même nom ne réutilise pas les identifiants de la précédente incarnation, SAUF si la nouvelle incarnation a une valeur différente de validité d'identifiant univoque. Voir la description de la commande UID pour les détails.

Le renommage INBOX est permis, et a un comportement particulier. Il déplace tous les messages dans INBOX vers une nouvelle boîte aux lettres avec le nom donné, laissant INBOX vide. Si la mise en œuvre de serveur accepte les noms hiérarchiques inférieurs de INBOX, ceux-ci ne sont pas affectés par un renommage de INBOX.

Exemples :

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

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

6.3.6 Commande SUBSCRIBE

Argument : boîte aux lettres

Réponse : pas de réponses spécifiques pour cette commande

Résultat : OK - subscribe achevé
 NO – échec de subscribe : pas d'abonnement possible à ce nom
 BAD – commande inconnue ou arguments invalides

La commande SUBSCRIBE ajoute le nom de boîte aux lettres spécifié à l'ensemble des boîtes aux lettres "actives" ou "souscrites" du serveur tel qu'il est retourné par la commande LSUB. Cette commande ne retourne une réponse OK étiquetée que si l'abonnement est réussi.

Un serveur PEUT valider l'argument de boîte aux lettres à SUBSCRIBE pour vérifier qu'elle existe. Cependant, il NE DOIT PAS retirer unilatéralement un nom de boîte aux lettres existant de la liste des abonnements même si il n'existe plus de boîte aux lettres de ce nom.

Note : Cette exigence découle de ce qu'un site de serveur peut choisir de retirer systématiquement une boîte aux lettres d'un nom bien connu (par exemple, "alertes-système") après l'arrivée à expiration de son contenu, dans l'intention de la recréer avec de nouveaux contenus le moment venu.

Exemple :

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

6.3.7 Commande UNSUBSCRIBE

Argument: nom de boîte aux lettres

Réponse : pas de réponses spécifiques pour cette commande

Résultat : OK - désabonnement achevé
NO – échec de désabonnement : pas possible de désabonner ce nom
BAD – commande inconnue ou arguments invalides

La commande UNSUBSCRIBE retire le nom de boîte aux lettres spécifié de l'ensemble des boîtes aux lettres "actives" ou "souscrites" du serveur tel que retourné par la commande LSUB. Cette commande ne retourne une réponse OK étiquetée que si le désabonnement est réussi.

Exemple :

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

6.3.8 Commande LIST

Arguments: nom de référence
nom de boîte aux lettres avec des caractères génériques possibles

Réponses : réponses non étiquetées : LIST

Résultat : OK - liste achevée
NO – échec de liste : cette référence ou nom ne peut être inscrit sur la liste
BAD – commande inconnue ou arguments invalides

La commande LIST retourne un sous-ensemble de noms pour l'ensemble complet de tous les noms disponibles pour le client. Zéro, une ou plusieurs réponses de LIST non étiquetées sont retournées, contenant les attributs des noms, les délimiteurs hiérarchiques, et les noms ; voir la description de la réponse LIST pour les détails.

La commande LIST DEVRAIT retourner ses données rapidement, sans délai inutile. Par exemple, elle NE DEVRAIT PAS se lancer à calculer l'état \Marked ou \Unmarked ou à effectuer d'autres traitements ; si chaque nom exige une seconde de traitement, une liste de 1200 noms prendrait alors 20 minutes !

Un argument de nom de référence vide (chaîne "") indique que le nom de boîte aux lettres est interprété comme par SELECT. Les noms de boîte aux lettres retournés DOIVENT correspondre au schéma de nom de boîte aux lettres fourni. Un argument de nom de référence non vide est le nom d'une boîte aux lettres ou d'un niveau de hiérarchie de boîte aux lettres, et indique le contexte dans lequel est interprété le nom de boîte aux lettres.

Un argument de nom de référence vide (chaîne "") est une demande spéciale de retourner le délimiteur hiérarchique et le nom racine du nom donné dans la référence. La valeur retournée comme racine PEUT être la chaîne vide si la référence n'est pas sur la racine ou est une chaîne vide. Dans tous les cas, un délimiteur hiérarchique (ou NIL si il n'y a pas de hiérarchie) est retourné. Cela permet à un client d'obtenir un délimiteur hiérarchique (ou de trouver que les noms de boîte aux lettres sont plats) même lorsque aucune boîte aux lettres de ce nom n'existe actuellement.

La référence et les arguments du nom de boîte aux lettres sont interprétés dans la forme canonique qui représente une hiérarchie de gauche à droite non ambiguë. Les noms de boîte aux lettres retournés seront dans la forme interprétée.

Note : L'interprétation de l'argument de référence est définie par la mise en œuvre. Elle dépend de ce que la mise en œuvre de serveur a le concept de "répertoire de travail en cours" et des "caractères de rupture" en tête, qui subrogent le répertoire de travail en cours.

Par exemple, sur un serveur qui exporte un système de fichiers UNIX ou NT, l'argument de référence contient le répertoire de travail en cours, et l'argument de nom de boîte aux lettres contiendrait le nom tel qu'interprété dans le répertoire de travail en cours.

Si une mise en œuvre de serveur n'a pas le concept de caractères de rupture, la forme canonique est normalement le nom de référence augmenté du nom de boîte aux lettres. Noter que si le serveur met en œuvre la convention d'espace de nom (paragraphe 5.1.2), "#" est un caractère de rupture et doit être traité comme tel.

Si l'argument de référence n'est pas un niveau de hiérarchie de boîte aux lettres (c'est-à-dire, que c'est un nom \NoInferiors), et/ou si l'argument de référence ne se termine pas par le délimiteur de hiérarchie, la façon de l'interpréter dépend de la mise en œuvre. Par exemple, une référence de "foo/bar" et un nom de boîte aux lettres de "rag/baz" pourraient être interprétés comme "foo/bar/rag/baz", "foo/barrag/baz", ou "foo/rag/baz". Un client NE DEVRAIT PAS utiliser de tels arguments de référence sauf à la demande explicite de l'utilisateur. Un navigateur hiérarchique NE DOIT PAS faire d'hypothèses sur l'interprétation de la référence par le serveur à moins que la référence soit un niveau de hiérarchie de boîte aux lettres ET se termine par le délimiteur hiérarchique.

Toute partie de l'argument de référence qui est incluse dans la forme interprétée DEVRAIT être en préfixe de la forme interprétée. Elle DEVRAIT aussi être sous la même forme que l'argument de nom de référence. Cette règle permet au client de déterminer si le nom de boîte aux lettres retourné est dans le contexte de l'argument de référence, ou si quelque chose relatif à l'argument de boîte aux lettres a subrogé l'argument de référence. Sans cette règle, le client devrait avoir connaissance de la sémantique de dénomination du serveur, y compris les caractères qui sont des "points de coupure" qui subrogent un contexte de dénomination.

Par exemple, voici comment des références et des noms de boîte aux lettres peuvent être interprétés sur un serveur 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 montrent des interprétations dans le contexte de l'argument de référence. Noter que "~smith/Mail" NE DEVRAIT PAS être transformé en quelque chose comme "/u2/users/smith/Mail", ou il serait impossible au client de déterminer que l'interprétation était la référence dans le contexte.

Le caractère "*" est un caractère générique, et correspond à zéro, un ou plusieurs caractères à cette position. Le caractère "%" est similaire à "*", mais il ne correspond pas à un délimiteur hiérarchique. Si le caractère générique "%" est le dernier caractère d'un argument de nom de boîte aux lettres, les niveaux correspondants de la hiérarchie sont aussi retournés. Si ces niveaux de hiérarchie ne sont pas aussi des boîtes aux lettres éligibles à la sélection, elles sont retournées avec l'attribut de nom de boîte aux lettres \Noselect (voir la description de la réponse LIST pour les détails).

Il est permis aux mises en œuvre de serveur de "cacher" des boîtes aux lettres autrement accessibles aux caractères génériques, en empêchant que certains caractères ou noms correspondent à un caractère générique dans certaines situations. Par exemple, un serveur UNIX peut interdire l'interprétation de "*" afin qu'un caractère initial "/" ne corresponde pas.

Le cas particulier de INBOX est inclus dans le résultat de LIST, si INBOX est accepté par ce serveur pour cet usager et si la chaîne majuscule "INBOX" correspond à la référence interprétée et aux arguments de nom de boîte aux lettres avec des caractères génériques comme décrit ci-dessus. Le critère pour omettre INBOX est que si SELECT INBOX va retourner un échec; il importe peu que le INBOX réel de l'usager réside sur ce serveur ou un autre.

Exemple :

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


6.3.9 Commande LSUB

Arguments : nom de référence
nom de boîte aux lettres avec de possibles caractères génériques

Réponses : réponses non étiquetées : LSUB

Résultat : OK – lsub achevé
NO – échec de lsub : pas possible d'afficher cette référence ou nom
BAD – commande inconnue ou arguments invalides

La commande LSUB retourne un sous-ensemble de noms à partir de l'ensemble des noms que l'utilisateur a déclarés comme étant "actifs" ou "souscrits". Zéro, une ou plusieurs réponses LSUB non étiquetées sont retournées. Les arguments pour LSUB sont de même forme que ceux pour LIST.

La réponse LSUB non étiquetée retournée PEUT contenir différents fanions de boîte aux lettres provenant d'une réponse LIST non étiquetée. Si cela devait arriver, les fanions dans la réponse LIST non étiquetée sont considérés comme d'autorité.

Une situation particulière survient lorsque on utilise LSUB avec le caractère générique %. Considérons ce qui arrive si "foo/bar" (avec un délimiteur hiérarchique de "/") est souscrit mais pas "foo". Un caractère générique "%" pour LSUB doit retourner foo, pas foo/bar, dans la réponse LSUB, et il DOIT être marqué avec l'attribut \Noselect.

Le serveur NE DOIT PAS retirer unilatéralement un nom de boîte aux lettres existant de la liste de souscription même si il n'existe plus de boîte aux lettres de ce nom.

Exemple :

```
C: A002 LSUB "#news." "comp.mail.*"
S: * LSUB () "." #news.comp.mail.mime
S: * LSUB () "." #news.comp.mail.misc
S: A002 OK LSUB achevé
C: A003 LSUB "#news." "comp.%"
S: * LSUB (\NoSelect) "." #news.comp.mail
S: A003 OK LSUB achevé
```

6.3.10 Commande STATUS

Arguments: nom de boîte aux lettres
noms d'éléments de données d'état

Réponses : réponses non étiquetées EXIGÉES : STATUS

Résultat : OK – état Achevé
NO – échec d'état : pas d'état pour ce nom
BAD – commande inconnue ou arguments invalides

La commande STATUS demande l'état de la boîte aux lettres indiquée. Il ne change pas la boîte aux lettres actuellement choisie, ni n'affecte l'état d'aucun message dans la boîte aux lettres interrogée (en particulier, STATUS NE DOIT PAS être cause que les messages perdent le fanion \Récent).

La commande STATUS fournit une solution de remplacement à l'ouverture d'une seconde connexion IMAP4rev1 et à l'exécution d'une commande EXAMINE sur une boîte aux lettres pour interroger l'état de cette boîte aux lettres sans désélectionner la boîte aux lettres en cours dans la première connexion IMAP4rev1.

À la différence de la commande LIST, la commande STATUS n'est pas d'une vitesse garantie dans sa réponse. Dans certaines circonstances, elle peut être assez lente. Dans certaines mises en œuvre, le serveur est obligé d'ouvrir la boîte aux lettres en lecture seule en interne pour obtenir certaines informations d'état. Également à la différence de la commande LIST, la commande STATUS n'accepte pas les caractères génériques.

Note : La commande STATUS est destinée à accéder à l'état des boîtes aux lettres autres que la boîte aux lettres actuellement sélectionnée. Parce que la commande STATUS peut causer l'ouverture interne de la boîte aux lettres, et parce que ces informations sont disponibles par d'autres moyens sur la boîte aux lettres choisie, la commande STATUS NE DEVRAIT PAS être utilisée sur la boîte aux lettres actuellement sélectionnée.

La commande STATUS NE DOIT PAS être utilisée comme opération de "vérification des nouveaux messages dans la boîte aux lettres sélectionnée" (voir aux paragraphes 7, 7.3.1, et 7.3.2 des informations complémentaires sur la méthode appropriée pour la vérification des nouveaux messages).

Parce qu'il n'est pas garanti que la commande STATUS soit rapide à obtenir des résultats, les clients NE DEVRAIENT PAS s'attendre à être capables de produire de nombreuses commandes STATUS consecutives et à obtenir des performances raisonnables.

Les éléments de données d'état actuellement définis qui peuvent être demandés sont :

MESSAGES : Le nombre de messages dans la boîte aux lettres.

RECENT : Le nombre de messages avec le fanion \Récent établi.

UIDNEXT : La prochaine valeur d'identifiant univoque de la boîte aux lettres. Voir le paragraphe 2.3.1.1 pour plus d'informations.

UIDVALIDITY : La valeur de validité de l'identifiant univoque de la boîte aux lettres. Voir le paragraphe 2.3.1.1 pour plus d'informations

UNSEEN : Le nombre de messages qui n'ont pas le fanion \Vu établi.

Exemple :

```
C: A042 STATUS blurdybloop (UIDNEXT MESSAGES)
S: * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)
S: A042 OK STATUS achevé
```

6.3.11 Commande APPEND

Arguments : nom de boîte aux lettres
Liste FACULTATIVE de fanions entre parenthèses
Chaîne de date/heure FACULTATIVE
message littéral

Réponses : pas de réponses spécifiques pour cette commande

Résultat : OK — ajout achevé
NO – erreur d'ajout : on ne peut pas ajouter à cette boîte aux lettres, erreur dans les fanions ou dans date/heure ou dans le texte du message.
BAD – commande inconnue ou arguments invalides

La commande APPEND ajoute l'argument littéral comme nouveau message à la fin de la boîte aux lettres de destination spécifiée. Cet argument DEVRAIT être dans le format d'un message de la [RFC-2822]. Les caractères à 8 bits sont permis dans le message. Une mise en œuvre de serveur qui n'est pas capable de préserver correctement les données à 8 bits DOIT être capable de convertir de façon réversible les données de APPEND en 8 bits en 7 bits en utilisant un codage de transfert de contenu MIME-IMB.

Note : Il PEUT y avoir des exceptions, par exemple, des brouillons de messages, dans lesquels les lignes d'en-tête exigées par la [RFC-2822] sont omises dans l'argument littéral du message à APPEND. Les pleines implications de cette façon d'agir DOIVENT être comprises et pesées avec soin.

Si une liste de fanions entre parenthèses est spécifiée, les fanions DEVRAIENT être établis dans le message résultant ; autrement, la liste des fanions du message résultant est réglée par défaut à vide. Dans tous les cas, le fanion Récent est aussi établi (*mis à 1*).

Si une date-heure est spécifiée, la date interne DEVRAIT être mise dans le message résultant ; autrement, la date interne du message résultant est réglée à la date et heure actuelle par défaut.

Si l'ajout échoue pour une raison quelconque, la boîte aux lettres DOIT être restaurée à son état antérieur à la tentative de APPEND ; aucun ajout partiel n'est permis.

Si la boîte aux lettres de destination n'existe pas, un serveur DOIT retourner une erreur, et NE DOIT PAS créer

automatiquement la boîte aux lettres. Sauf si il est certain que la boîte aux lettres de destination ne peut pas être créée, le serveur DOIT envoyer le code de réponse "[TRYCREATE]" comme préfixe du texte de la réponse NO étiquetée. Cela donne au client l'indication qu'il peut essayer une commande CREATE et réessayer la commande APPEND si CREATE réussit.

Si la boîte aux lettres est celle actuellement choisie, les actions normales de nouveau message DEVRAIENT se produire. Précisément, le serveur DEVRAIT le notifier immédiatement au client via une réponse EXISTS non étiquetée. Si le serveur ne le fait pas, le client PEUT produire une commande NOOP (ou à défaut, une commande CHECK) après une ou plusieurs commandes APPEND.

Exemple :

```
C: A003 APPEND messages sauvegardés (\Vus) {310}
S: + Prêt pour des données littérales
C: Date: lundi 7 février 1994 21:52:25 -0800 (PST)
C: From: Fred Foobar <foobar@Blurdybloop.COM>
C: Sujet: réunion d'après midi
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: Bonjour Joe, peut-on se réunir demain à 3:30 ?
C:
S: A003 OK APPEND achevé
```

Note : La commande APPEND n'est pas utilisée pour la livraison de message, parce qu'elle ne fournit pas de mécanisme pour transférer les informations d'enveloppe SMTP.

6.4 Commandes du client – état Choisi

Dans l'état Choisi, les commandes qui manipulent les messages dans une boîte aux lettres sont permises.

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

6.4.1 Commande CHECK

Argument : aucun

Réponse : pas de réponses spécifiques pour cette commande

Résultat : OK – vérification achevée
BAD – commande inconnue ou arguments invalides

La commande CHECK demande un point de vérification de la boîte aux lettres actuellement choisie. Un point de vérification se réfère à tout entretien dépendant de la mise en œuvre associé à la boîte aux lettres (par exemple, alignement de l'état de mémoire de la boîte aux lettres chez le serveur avec l'état sur le disque) qui n'est normalement pas exécuté au titre de chaque commande. Un point de vérification PEUT prendre une quantité de temps réel non instantanée pour se mener à terme. Si une mise en œuvre de serveur n'a pas ces considérations d'entretien, CHECK est équivalent à NOOP.

Il n'est pas garanti qu'une réponse EXISTS non étiquetée va arriver par suite de CHECK. NOOP, et non pas CHECK, DEVRAIT être utilisé pour l'interrogation des nouveaux messages.

Exemple :

```
C: FXXZ CHECK
S: FXXZ OK CHECK achevé
```

6.4.2 Commande CLOSE

Arguments : aucun

Réponses : pas de réponses spécifiques pour cette commande

Résultat : OK – fermeture achevée, maintenant dans l'état authentifié
BAD – commande inconnue ou arguments invalides

La commande CLOSE retire de façon permanente tous les messages qui ont le fanion \Supprimé établi de la boîte aux lettres actuellement choisie, et retourne à l'état Authentifié à partir de l'état Choisi. Aucune réponse EXPUNGE non étiquetée n'est envoyée.

Aucun message n'est retiré, et aucune erreur n'est générée si la boîte aux lettres est choisie par une commande EXAMINE ou est par ailleurs sélectionnée en lecture seule.

Même si une boîte aux lettres est choisie, une commande SELECT, EXAMINE ou LOGOUT PEUT être produite sans avoir précédemment produit une commande CLOSE. Les commandes SELECT, EXAMINE et LOGOUT closent implicitement la boîte aux lettres actuellement choisie sans faire un effacement. Cependant, lorsque beaucoup de messages sont supprimés, une séquence CLOSE-LOGOUT ou CLOSE-SELECT est considérablement plus rapide qu'une EXPUNGE-LOGOUT ou EXPUNGE-SELECT parce qu'aucune réponse EXPUNGE non étiquetée (que le client ignorerait probablement) n'est envoyée.

Exemple :

C: A341 CLOSE

S: A341 OK CLOSE achevé

6.4.3 Commande EXPUNGE

Arguments : aucun

Réponses : réponses non étiquetées : EXPUNGE

Résultat : OK – effacement achevé
NO – échec d'effacement : on ne peut pas effacer (par exemple, permission refusée)
BAD – commande inconnue ou arguments invalides

La commande EXPUNGE efface de façon permanente tous les messages qui ont le fanion \Supprimé établi sur la boîte aux lettres actuellement choisie. Avant de retourner un OK au client, une réponse EXPUNGE non étiquetée est envoyée pour chaque message retiré. Noter que si un message avec le fanion \Récent établi est retiré, une réponse RECENT non étiquetée est envoyée après le ou les EXPUNGE non étiquetés pour mettre à jour le compte des messages RECENT du client.

Exemple :

C: A202 EXPUNGE

S: * 3 EXPUNGE

S: * 3 EXPUNGE

S: * 5 EXPUNGE

S: * 8 EXPUNGE

S: A202 OK EXPUNGE achevé

Note : Dans cet exemple, les messages 3, 4, 7 et 11 avaient le fanion \Supprimé établi. Voir la description de la réponse EXPUNGE pour des explications complémentaires.

6.4.4 Commande SEARCH

Arguments : spécification FACULTATIVE du CHARSET (*jeu de caractère*)
critères de recherche (un ou plusieurs)

Réponses : réponse non étiquetée EXIGÉE : SEARCH

Résultat : OK – recherche achevée
NO – erreur de recherche : on ne peut pas rechercher ce CHARSET ou critère
BAD – commande inconnue ou arguments invalides

La commande SEARCH recherche dans la boîte aux lettres les messages qui correspondent aux critères de recherche donnés. Les critères de recherche consistent en une ou plusieurs clés de recherche. La réponse SEARCH non étiquetée du serveur contient une liste des numéros de séquence de messages correspondant aux messages qui satisfont aux critères de recherche.

Lorsque plusieurs clés sont spécifiées, le résultat est l'intersection (fonction ET) de tous les messages qui correspondent à ces clés. Par exemple, le critère SUPPRIMÉ DE "SMITH" DEPUIS LE 1-FEB-1994 se réfère à tous les messages supprimés de Smith qui ont été placés dans la boîte aux lettres depuis le 1^{er} février 1994. Une clé de recherche peut aussi être une liste entre parenthèses d'une ou plusieurs clés de recherche (par exemple, à utiliser avec les clés OU et NON).

Les mises en œuvre de serveur PEUVENT exclure les parties de corps MIME-IMB avec des types de support de contenu terminal autres que TEXT et MESSAGE de la prise en considération dans une correspondance SEARCH.

La spécification FACULTATIVE du CHARSET comporte le mot "CHARSET" suivi par un jeu de caractères enregistré. Il indique le jeu de caractères des chaînes qui apparaissent dans les critères de recherche. Les codages de transfert de contenu MIME-IMB et les chaînes MIME-HDRS [RFC2047] dans les en-têtes de la [RFC2822]/[RFC2045] DOIVENT être décodés avant de comparer le texte. L'US-ASCII DOIT être pris en charge ; les autres jeux de caractères PEUVENT être acceptés.

Si le serveur ne prend pas en charge le jeu de caractères spécifié, il DOIT retourner une réponse NO (et non pas BAD) étiquetée. Cette réponse DEVRAIT contenir le code de réponse BADCHARSET, qui PEUT faire la liste des jeux de caractères pris en charge par le serveur.

Dans toutes les clés de recherche qui utilisent des chaînes, un message correspond à la clé si la chaîne est une sous-chaîne du texte associé. La correspondance est insensible à la casse. Noter que la chaîne vide est une sous-chaîne ; ceci est utile lors d'une recherche de HEADER (*en-tête*).

Les clés de recherche définies sont les suivantes. Se référer à la Section 9 "Syntaxe formelle" pour les définitions syntaxiques précises des arguments.

<ensemble de séquence>

Messages avec le numéro de séquence de message correspondant à l'ensemble spécifié de numéros de séquence de message.

ALL

Tous les messages de la boîte aux lettres ; la clé initiale par défaut pour l'opération ET.

ANSWERED

Messages avec le fanion \Répondu mis.

BCC <chaîne>

Messages qui contiennent la chaîne spécifiée dans le champ BCC de la structure d'enveloppe.

BEFORE <date>

Messages dont la date interne (sans considération de l'heure et de la zone horaire) est plus tôt que la date spécifiée.

BODY <chaîne>

Messages qui contiennent la chaîne spécifiée dans le corps du message.

CC <chaîne>

Messages qui contiennent la chaîne spécifiée dans le champ CC de la structure d'enveloppe.

DELETED

Messages dont le fanion \Supprimé est établi.

DRAFT

Messages dont le fanion \Brouillon est établi.

FLAGGED

Messages dont le fanion \Fanion est établi.

FROM <chaîne>

Messages qui contiennent la chaîne spécifiée dans le champ FROM de la structure d'enveloppe.

HEADER <nom-de-champ> <chaîne>

Messages qui ont un en-tête avec le champ-nom spécifié (comme défini dans la [RFC2822]) et qui contiennent la chaîne spécifiée dans le texte de l'en-tête (ce qui vient après les deux-points). Si la chaîne à rechercher est de longueur nulle, cela correspond à tous les messages qui ont une ligne d'en-tête avec le nom-de-champ spécifié, sans considération du contenu.

KEYWORD <fanion>

Messages qui ont le fanion de mot-clé spécifié.

LARGER <n>

Messages d'une taille de la [RFC2822] supérieure à nombre d'octets spécifié.

NEW

Messages qui ont le fanion \Récent établi mais pas le fanion \Vu. Cela est fonctionnellement équivalent à "(RECENT UNSEEN)".

NOT <clé-de-recherche>

Messages qui ne correspondent pas à la clé de recherche spécifiée.

OLD

Messages qui n'ont pas le fanion \Récent établi. Cela est fonctionnellement équivalent à "NOT RECENT" (par opposition à "NOT NEW").

ON <date>

Messages dont la date interne (sans considération de l'heure et de la zone horaire) est dans la date spécifiée.

OR <clé-de-recherche1> <clé-de-recherche2>

Messages qui correspondent à l'une ou l'autre des clés de recherche.

RECENT

Messages qui ont le fanion \Récent établi.

SEEN

Messages qui ont le fanion \Vu établi.

SENTBEFORE <date>

Messages dont l'en-tête Date: de la [RFC2822] (sans considération de l'heure et de la zone horaire) est plus tôt que la date spécifiée.

SENTON <date>

Messages dont l'en-tête Date: de la [RFC2822] (sans considération de l'heure et de la zone horaire) est dans la date spécifiée.

SENTSINCE <date> Messages dont l'en-tête Date: de la [RFC2822] (sans considération de l'heure et de la zone horaire) est dans la date spécifiée ou plus tard.

SINCE <date>

Messages dont la date interne (sans considération de l'heure et de la zone horaire) est dans la date spécifiée ou plus tard.

SMALLER <n>

Messages avec une taille selon la [RFC2822] inférieure au nombre d'octets spécifié.

SUBJECT <chaîne>

Messages qui contiennent la chaîne spécifiée dans le champ SUBJECT de la structure d'enveloppe.

TEXT <chaîne>

Messages qui contiennent la chaîne spécifiée dans l'en-tête ou corps de message.

TO <chaîne>

Messages qui contiennent la chaîne spécifiée dans le champ TO de la structure d'enveloppe.

UID <ensemble de séquence>

Messages avec des identifiants univoques correspondant à l'ensemble spécifié d'identifiants univoques. Les gammes d'ensemble de séquences sont permises.

UNANSWERED

Messages qui n'ont pas le fanion \Répondu établi.

UNDELETED

Messages qui n'ont pas le fanion \Supprimé établi.

UNDRAFT

Messages qui n'ont pas le fanion \Brouillon établi.

UNFLAGGED

Messages qui n'ont pas le fanion \Fanion établi.

UNKEYWORD <fanion>

Messages qui n'ont pas le fanion de mot-clé spécifié établi.

UNSEEN

Messages qui n'ont pas le fanion \Vu établi.

Exemple :

```
C: A282 SEARCH FLAGGED SINCE 1-Feb-1994 NOT FROM "Smith"
S: * SEARCH 2 84 882
S: A282 OK SEARCH achevé
C: A283 SEARCH TEXT "cette chaîne n'est pas dans la boîte aux lettres"
S: * SEARCH
S: A283 OK SEARCH achevé
C: A284 SEARCH CHARSET UTF-8 TEXT {6}
S: + Prêt pour du texte littéral
C: XXXXXX
S: * SEARCH 43
S: A284 OK SEARCH achevé
```

Note : Comme le présent document se restreint au texte ASCII à 7 bits, il n'est pas possible de montrer des données UTF-8 réelles. Le "XXXXXX" est un fourre-tout pour ce qui pourrait être 6 octets de données à 8 bits dans une transaction réelle.

6.4.5 Commande FETCH

Arguments: ensemble de séquences
noms ou macro d'éléments de données de message

Réponses : réponses non étiquetées : FETCH

Résultat : OK – fetch achevé
NO – erreur de collecte : on ne peut pas aller chercher ces données
BAD – commande inconnue ou arguments invalides

La commande FETCH restitue les données associées à un message de la boîte aux lettres. Les éléments de données à aller chercher peuvent être un seul atome ou une liste entre parenthèses.

La plupart des éléments de données, identifiés dans la syntaxe formelle de la règle msg-att-static, sont statiques et NE DOIVENT PAS changer pour un message particulier. Les autres éléments de données, identifiés en syntaxe formelle sous la règle msg-att-dynamic, PEUVENT changer, soit par suite d'une commande STORE, soit par suite d'événements externes.

Par exemple, si un client reçoit une ENVELOPE pour un message alors qu'il connaît déjà l'enveloppe, il peut en toute sécurité ignorer l'enveloppe nouvellement transmise.

Il y a trois macros qui spécifient les ensembles d'éléments de données communément utilisés, et peuvent être utilisés à la place des éléments de données. Une macro doit être utilisée par elle-même, et non en conjonction avec d'autres macros ou éléments de données.

ALL

Macro équivalente à : (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)

FAST

Macro équivalente à : (FLAGS INTERNALDATE RFC822.SIZE)

FULL

Macro équivalente à : (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY)

Les éléments de données actuellement définis qu'on peut aller chercher sont :

BODY

Forme non extensible de BODYSTRUCTURE.

BODY[<section>]<<partial>>

Le texte d'une section de corps particulière. La spécification de la section est un ensemble de zéro, un ou plusieurs spécificateurs de partie délimités par des points. Un spécificateur de partie est soit un numéro de partie soit un des éléments suivants : HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, et TEXT. Une spécification de section vide se réfère au message entier, y compris l'en-tête.

Chaque message a au moins un numéro de partie . Les messages non MIME-IMB, et non MIME-IMB multipartie sans message encapsulé n'ont qu'une partie 1.

Les messages multiparties ont des numéros de parties consécutifs, comme elles surviennent dans le message. Si une partie particulière est du type message ou multipart, ses parties DOIVENT être indiquées par un point suivi par le numéro de partie au sein de la partie multipartie qui l'incorpore.

Une partie du type MESSAGE/RFC822 a aussi des numéros de parties incorporées, qui se réfèrent aux parties du corps de la partie MESSAGE.

Les spécificateurs de parties HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, et TEXT peuvent être le seul spécificateur de partie ou bien ils peuvent être précédés d'un préfixe par un ou plusieurs spécificateurs de partie numériques, pourvu que le spécificateur de partie numérique se réfère à une partie de type MESSAGE/RFC822. Le spécificateur de partie MIME DOIT avoir en préfixe un ou plusieurs spécificateurs de partie numériques.

Les spécificateurs de parties HEADER, HEADER.FIELDS, et HEADER.FIELDS.NOT se réfèrent à l'en-tête de message de la [RFC2822] ou à un message MESSAGE/RFC822 [RFC2046] encapsulé. HEADER.FIELDS et HEADER.FIELDS.NOT sont suivis par une liste de noms de champs (comme défini dans la [RFC2822]), et retournent un sous-ensemble de l'en-tête. Le sous-ensemble retourné par HEADER.FIELDS contient seulement les champs d'en-tête qui ont un nom de champ qui correspond à un des noms de la liste ; de même, le sous-ensemble retourné par HEADER.FIELDS.NOT contient seulement les champs d'en-tête qui ont un nom de champ qui ne correspond pas. La correspondance de champ est insensible à la casse mais exacte autrement. La répartition en sous-ensembles n'exclut pas la ligne blanche de délimitation de la [RFC2822] entre l'en-tête et le corps ; la ligne blanche est incluse dans toutes les collectes d'en-têtes, excepté dans le cas d'un message qui n'a ni corps ni ligne blanche.

Le spécificateur de partie MIME se réfère à l'en-tête MIME-IMB pour cette partie.

Le spécificateur de partie TEXT se réfère au corps de texte du message, omettant l'en-tête de la [RFC2822].

Voici un exemple de message complexe avec quelques un de ses spécificateurs de parties :

```

HEADER (En-tête [RFC2822] du message)
TEXT (Corps de texte [RFC2822] du message) MULTIPART/MIXED
1 TEXT/PLAIN
2 APPLICATION/OCTET-STREAM
3 MESSAGE/RFC822
3. HEADER (En-tête [RFC2822] du message)
3. TEXT (Corps de texte [RFC2822] du message) MULTIPART/MIXED
3.1 TEXT/PLAIN
3.2 APPLICATION/OCTET-STREAM
4 MULTIPART/MIXED
4.1 IMAGE/GIF
4.1 MIME (En-tête MIME-IMB pour une IMAGE/GIF)
4.2 MESSAGE/RFC822

```


4.2 HEADER (En-tête [RFC2822] du message)
 4.2 TEXT (Corps de texte [RFC2822] du message) MULTIPART/MIXED
 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 d'aller chercher une sous-chaîne du texte désigné. Cela se fait en apposant un signe inférieur à (" $<$ "), la position d'octet du premier octet désiré, un point, le nombre maximum d'octets désiré, et un signe supérieur à (" $>$ ") au spécificateur de partie. Si l'octet de début est au delà de la fin du texte, une chaîne vide est retournée.

Toute recherche partielle qui tente de lire au delà de la fin du texte est tronquée comme approprié. Une recherche partielle qui commence à l'octet 0 est retournée comme une recherche partielle, même si cette troncature survient.

Note : Cela signifie que BODY[$<$ 0.2048 $>$] d'un message de 1500 octets va retourner BODY[$<$ 0 $>$] avec une partie littérale de taille 1500, et non de BODY[$<$].

Note : La recherche d'une sous-chaîne d'un spécificateur de partie de HEADER.FIELDS ou HEADER.FIELDS.NOT est calculée après avoir subdivisé l'en-tête.

Le fanion \Vu est implicitement établi ; si cela cause le changement des fanions, ils DEVRAIENT être inclus au titre des réponses FETCH.

BODY.PEEK[$<$ section $>$][$<<$ partial $>>$]

Une forme de remplacement de BODY[$<$ section $>$] qui n'établit pas implicitement le fanion \Vu.

BODYSTRUCTURE

Structure du corps MIME-IMB du message. Ceci est calculé par le serveur en analysant les champs d'en-tête MIME-IMB dans l'en-tête de la [RFC2822] et les en-têtes MIME-IMB.

ENVELOPE

La structure d'enveloppe du message. Ceci est calculé par le serveur en analysant l'en-tête de la [RFC2822] dans les parties composantes, en prenant divers champs par défaut en tant que de besoin.

FLAGS

Les fanions qui sont établis pour ce message.

INTERNALDATE

La date interne du message.

RFC822

Fonctionnellement équivalent à BODY[$<$], différant par la syntaxe des données FETCH non étiquetées résultantes (RFC822 est retourné).

RFC822.HEADER

Fonctionnellement équivalent à BODY.PEEK[HEADER], différant par la syntaxe des données FETCH non étiquetées résultantes (RFC822.HEADER est retourné).

RFC822.SIZE

La taille du message selon la [RFC2822].

RFC822.TEXT

Fonctionnellement équivalent à BODY[TEXT], différant par la syntaxe des données FETCH non étiquetées résultantes (RFC822.TEXT est retourné).

UID

L'identifiant univoque pour le message.

Exemple :

C: A654 FETCH 2:4 (FLAGS BODY[HEADER.FIELDS (DATE FROM)])

S: * 2 FETCH

S: * 3 FETCH

S: * 4 FETCH

S: A654 OK FETCH achevé

6.4.6 Commande STORE

Arguments : ensemble de séquences
 nom d'éléments de données de message
 valeur pour les éléments de données de message

Réponses : réponses non étiquetées : FETCH

Résultat : OK – mémorisation achevée
 NO – erreur de mémorisation : on ne peut pas mémoriser ces données
 BAD – commande inconnue ou arguments invalides

La commande STORE altère les données associées à un message dans la boîte aux lettres. Normalement, STORE va retourner la valeur mise à jour des données avec une réponse FETCH non étiquetée. Un suffixe de ".SILENT" dans le nom d'élément de données empêche le FETCH non étiqueté, et le serveur DEVRAIT supposer que le client a déterminé lui-même la valeur mise à jour, ou ne se soucie pas de la mise à jour de la valeur.

Note : Sans considération de l'utilisation éventuelle du suffixe ".SILENT", le serveur DEVRAIT envoyer une réponse FETCH non étiquetée si un changement des fanions d'un message provenant d'une source externe est observé. L'intention est que l'état des fanions soit déterminé sans compétition.

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

FLAGS <liste-de-fanions>

Remplace les fanions pour le message (autres que \Récent) par l'argument. La nouvelle valeur des fanions est retournée comme si un FETCH de ces fanions était effectué.

FLAGS.SILENT <liste-de-fanions>

Équivalent à FLAGS, mais sans retourner une nouvelle valeur.

+FLAGS <liste-de-fanions>

Ajoute l'argument aux fanions pour le message. La nouvelle valeur des fanions est retournée comme si un FETCH de ces fanions était effectué.

+FLAGS.SILENT <liste-de-fanions>

Équivalent à +FLAGS, mais sans retourner une nouvelle valeur.

-FLAGS <liste-de-fanions>

Retire l'argument des fanions pour le message. La nouvelle valeur des fanions est retournée comme si un FETCH de ces fanions était effectué.

-FLAGS.SILENT <liste-de-fanions>

Équivalent à -FLAGS, mais sans retourner une nouvelle valeur.

Exemple :

```
C: A003 STORE 2:4 +FLAGS (\Supprimé)
S: * 2 FETCH (FLAGS (\Supprimé \Vu))
S: * 3 FETCH (FLAGS (\Supprimé))
S: * 4 FETCH (FLAGS (\Supprimé \Fanion \Vu))
S: A003 OK STORE achevé
```

6.4.7 Commande COPY

Arguments : ensemble de séquences
 nom de boîte aux lettres

Réponse : pas de réponses spécifiques pour cette commande

Résultat : OK – copie achevée
 NO – erreur de copie : on ne peut pas copier ces messages ou à ce nom
 BAD – commande inconnue ou arguments invalides

La commande COPY copie le ou les messages spécifiés à la fin de la boîte aux lettres de destination spécifiée. Les fanions et la date interne du ou des messages DEVRAIENT être préservés, et le fanion Récent DEVRAIT être établi dans la copie.

Si la boîte aux lettres de destination n'existe pas, un serveur DEVRAIT retourner une erreur. Il NE DEVRAIT PAS créer automatiquement la boîte aux lettres. Sauf si il est certain que la boîte aux lettres de destination ne peut pas être créée, le serveur DOIT envoyer le code de réponse "[TRYCREATE]" comme préfixe du texte de la réponse NO étiquetée. Cela donne au client l'indication qu'il peut tenter une commande CREATE et réessayer la COPY si la CREATE réussit.

Si la commande COPY ne réussit pas pour une raison quelconque, une mise en œuvre de serveur DOIT restaurer la boîte aux lettres de destination dans l'état antérieur à la tentative de COPY.

Exemple :

```
C: A003 COPY 2:4 MEETING
S: A003 OK COPY achevé
```

6.4.8 Commande UID

Arguments : nom de commande
arguments de commande

Réponses : réponses non étiquetées : FETCH, SEARCH

Résultat : OK – commande UID achevée
NO – erreur de commande UID
BAD – commande inconnue ou arguments invalides

La commande UID a deux formes. Sous la première forme, elle prend comme arguments une commande COPY, FETCH, ou STORE avec des arguments appropriés pour la commande associée. Cependant, les numéros dans l'argument d'ensemble de séquence sont des identifiants univoques au lieu du numéro de séquence du message. Les gammes d'ensemble de séquence sont permises, mais il n'est pas garanti que les identifiants univoques seront contigus.

Un identifiant univoque non existant est ignoré sans qu'aucun message d'erreur soit généré. Et donc, il est possible qu'une commande UID FETCH retourne un OK sans aucune donnée ou qu'une UID COPY ou UID STORE retourne un OK sans effectuer aucune opération.

Dans la seconde forme, la commande UID prend une commande SEARCH avec les arguments de commande SEARCH. L'interprétation des arguments est la même qu'avec SEARCH ; cependant, les numéros retournés dans une réponse SEARCH à une commande UID SEARCH sont les identifiants univoques au lieu des numéros de séquence de message. Par exemple, la commande UID SEARCH 1:100 UID 443:557 retourne les identifiants univoques correspondant à l'intersection des deux ensembles de séquence, la gamme de numéros de séquence de message de 1 à 100 et la gamme d'UID de 443 à 557.

Note : dans l'exemple ci-dessus apparaît la gamme des UID de 443 à 557. Le commentaire sur l'identifiant univoque non existant qui est ignoré sans aucun message d'erreur s'applique ici aussi. Et donc, même si l'UID 443 ni 557 n'existent, cette gamme est valide et inclurait un UID 495 existant.

Noter aussi qu'une gamme d'UID de 559:* inclut toujours l'UID du dernier message de la boîte aux lettres, même si 559 est supérieur à toute valeur d'UID allouée. Cela parce que le contenu d'une gamme est indépendant de l'ordre des points d'extrémité de la gamme. Et donc, toute gamme d'UID avec une * a un de ses points d'extrémité qui indique au moins un message (le message avec l'UID de numéro le plus élevé), sauf si la boîte aux lettres est vide.

Le nombre après "*" dans une réponse FETCH non étiquetée est toujours un numéro de séquence de message, et non un identifiant univoque, même pour une réponse à une commande UID. Cependant, les mises en œuvre de serveur DOIVENT implicitement inclure l'élément de données de message UID au titre de toute réponse FETCH causée par une commande UID, sans considération du fait qu'un UID était ou non spécifié comme élément de données de message pour FETCH.

Note : La règle d'inclusion de l'élément de données de message UID au titre d'une réponse FETCH s'applique principalement aux commandes UID FETCH et UID STORE, y compris une commande UID FETCH qui ne comporte pas d'UID comme élément de données de message. Bien qu'il soit peu vraisemblable que les autres commandes UID causent un FETCH non étiqueté, cette règle s'applique également à ces commandes.

Exemple :

```
C: A999 UID FETCH 4827313:4828442 FLAGS
S: * 23 FETCH (FLAGS (\Vu) UID 4827313)
S: * 24 FETCH (FLAGS (\Vu) UID 4827943)
S: * 25 FETCH (FLAGS (\Vu) UID 4828442)
S: A999 OK UID FETCH achevé
```

6.5 Commandes du client – Expérimental/Expansion

6.5.1 Commande X<atom>

Argument : défini par la mise en œuvre

Réponse : défini par la mise en œuvre

Résultat : OK – commande achevée
NO – échec
BAD – commande inconnue ou arguments invalides

Toute commande préfixée d'un X est une commande expérimentale. Les commandes qui ne font pas partie de la présente spécification, une révision normalisée ou en cours de normalisation de la présente spécification, ou un protocole expérimental approuvé de l'IESG, DOIVENT utiliser le préfixe X.

Toutes les réponses non étiquetées ajoutées produites par une commande expérimentale DOIVENT aussi être préfixées d'un X. Les mises en œuvre de serveur NE DOIVENT PAS envoyer de telles réponses non étiquetées, à moins que le client ne le demande en produisant la commande expérimentale associée.

Exemple :

```
C: a441 CAPABILITY
S: * CAPABILITY IMAP4rev1 XPIG-LATIN
S: a441 OK CAPABILITY achevé
C: A442 XPIG-LATIN
S: * XPIG-LATIN ow-nay eaking-spay ig-pay atin-lay
S: A442 OK XPIG-LATIN ompleted-cay
```

7. Réponses du serveur

Les réponses du serveur sont de trois formes : réponses d'état, données du serveur, et demande de continuation de commande. Les informations contenues dans une réponse de serveur, identifiées par "Contents:" dans les descriptions de réponse ci-dessous, sont décrites par fonction, et non par syntaxe. La syntaxe précise des réponses de serveur est décrite dans la section 9 "Syntaxe formelle".

Le client DOIT être prêt à accepter toutes réponses à tout moment.

Les réponses d'état peuvent être étiquetées ou non étiquetées. Les réponses d'état étiquetées indiquent le résultat d'achèvement (état OK, NO ou BAD) d'une commande du client, et ont une étiquette correspondant à la commande.

Certaines réponses d'état, et toutes les données du serveur, sont non étiquetées. Une réponse non étiquetée est indiquée par le jeton "*" à la place d'une étiquette. Les réponses d'état non étiquetées indiquent l'accueil du serveur, ou l'état du serveur qui n'indique pas l'achèvement d'une commande (par exemple, une alerte de fermeture imminente du système). Pour des raisons historiques, les réponses de données du serveur non étiquetées sont aussi appelées "données non sollicitées", bien que strictement parlant, seules les données unilatérales du serveur soient vraiment "non sollicitées".

Certaines données du serveur DOIVENT être enregistrées par le client lorsque elles sont reçues ; cela est noté dans la description de ces données. De telles données portent des informations critiques qui affectent l'interprétation de toutes les commandes et réponses suivantes (par exemple, mises à jour reflétant la création ou l'effacement de messages).

Les autres données du serveur DEVRAIENT être enregistrées pour référence ultérieure ; si le client n'a pas besoin d'enregistrer les données, ou si l'enregistrement des données n'a pas d'objet évident (par exemple, une réponse SEARCH lorsque aucune commande SEARCH n'est en cours) les données DEVRAIENT être ignorées.

Un exemple de données unilatérales non étiquetées du serveur survient lorsque la connexion IMAP est dans l'état choisi. Dans l'état choisi, le serveur vérifie si la boîte aux lettres a de nouveaux messages au titre de l'exécution d'une commande. Normalement, cela fait partie de l'exécution de chaque commande ; et donc, une commande NOOP suffit à vérifier les nouveaux messages. Si de nouveaux messages sont trouvés, le serveur envoie des réponses EXISTS et RECENT non étiquetées qui reflètent la nouvelle taille de la boîte aux lettres. Les mises en œuvre de serveur qui offrent plusieurs accès simultanés à la même boîte aux lettres DEVRAIENT aussi envoyer des réponses FETCH et EXPUNGE unilatérales non étiquetées appropriées si un autre agent change l'état de tout fanion de message ou efface des messages.

Les réponses de demande de continuation de commande utilisent le jeton "+" à la place d'une étiquette. Ces réponses sont envoyées par le serveur pour indiquer l'acceptation d'une commande incomplète du client et qu'il est prêt pour le reste de la commande.

7.1 Réponses du serveur – Réponses d'état

Les réponses d'état sont OK, NO, BAD, PREAUTH et BYE. OK, NO et BAD peuvent être étiquetées ou non étiquetées. PREAUTH et BYE sont toujours non étiquetées.

Les réponses d'état PEUVENT inclure un "code de réponse" FACULTATIF. Un code de réponse consiste en données entre des crochets sous la forme d'un atome, éventuellement suivi par une espace et des arguments. Les codes de réponse contiennent des informations supplémentaires ou des codes d'état pour le logiciel client au delà des conditions OK/NO/BAD, et sont définis lorsque une action spécifique peut être effectuée par un client sur la base de ces informations.

Les codes de réponse actuellement définis sont :

ALERT

Le texte lisible par l'homme contient une alerte spéciale qui DOIT être présentée à l'utilisateur d'une façon qui attire son attention sur le message.

BADCHARSET

Facultativement suivi par une liste de jeux de caractères entre parenthèses. Une commande SEARCH a échoué parce que le jeu de caractères donné n'est pas accepté par cette mise en œuvre. Si la liste facultative des jeux de caractères est donnée, elle fait la liste des jeux de caractères qui sont pris en charge par cette mise en œuvre.

CAPABILITY

Suivi par une liste de capacités. Cela peut apparaître dans le OK initial ou dans la réponse PREAUTH à la transmission d'une liste initiale de capacités. Cela rend inutile l'envoi par un client d'une commande CAPABILITY séparée si il reconnaît cette réponse.

PARSE

Le texte lisible par l'homme représente une erreur dans l'analyse de l'en-tête [RFC-2822] ou les en-têtes MIME-IMB d'un message de la boîte aux lettres.

PERMANENTFLAGS

Suivi par une liste de fanions entre parenthèses, indique lesquels des fanions connus le client peut changer de façon permanente. Un fanion qui est dans la réponse FLAGS non étiquetée, mais pas dans la liste PERMANENTFLAGS, ne peut pas être établi de façon permanente. Si le client tente STORE sur un fanion qui n'est pas dans la liste des PERMANENTFLAGS, le serveur ignorera le changement ou mémorisera le changement d'état seulement pour le reste de la session en cours. La liste PERMANENTFLAGS peut aussi inclure le fanion spécial *, qui indique qu'il est possible de créer de nouveaux mots-clés en essayant de mémoriser ces fanions dans la boîte aux lettres.

READ-ONLY

La boîte aux lettres est sélectionnée en lecture seule, ou son accès lors de la sélection est passé de lecture-écriture à lecture seule.

READ-WRITE

La boîte aux lettres est sélectionnée en lecture-écriture, ou son accès lors de la sélection a été changé de lecture seule à lecture-écriture.

TRYCREATE

Une tentative de APPEND ou COPY échoue parce que la boîte aux lettres cible n'existe pas (par opposition à d'autres

raisons). C'est l'indication au client que cette opération peut réussir si la boîte aux lettres est d'abord créée par la commande CREATE.

UIDNEXT

Suivi par un nombre décimal, il indique la prochaine valeur d'identifiant univoque. Se reporter au paragraphe 2.3.1.1 pour plus d'informations.

UIDVALIDITY

Suivi par un nombre décimal, il indique la valeur de validité de l'identifiant univoque. Se reporter au paragraphe 2.3.1.1 pour plus d'informations.

UNSEEN

Suivi par un nombre décimal, il indique le numéro du premier message sans le fanion \Vu établi.

Des codes de réponse supplémentaires définis par un client ou mise en œuvre de serveur particulier DEVRAIENT avoir en préfixe un "X" jusqu'à ce qu'ils soient ajoutés à une révision de ce protocole. Les mises en œuvre de client DEVRAIENT ignorer les codes de réponse qu'elles ne reconnaissent pas.

7.1.1 Réponse OK

Contenu: code de réponse FACULTATIF
texte lisible par l'homme

La réponse OK indique un message d'information de la part du serveur. Lorsque elle est étiquetée, elle indique l'achèvement réussi de la commande associée. Le texte lisible par l'homme PEUT être présenté à l'utilisateur comme un message d'information. La forme non étiquetée indique un message seulement pour information ; la nature de l'information PEUT être indiquée par un code de réponse.

La forme non étiquetée est aussi utilisée comme un des trois accueils possibles au début de la connexion. Elle indique que la connexion n'est pas encore authentifiée et qu'une commande LOGIN est nécessaire.

Exemple :

```
S: * OK IMAP4rev1 serveur prêt
C: A001 LOGIN fred blurrybloop
S: * OK [ALERT] Coupure système dans 10 minutes
S: A001 OK LOGIN achevé
```

7.1.2 Réponse NO

Contenu : code de réponse FACULTATIF
texte lisible par l'homme

La réponse NO indique un message d'erreur de fonctionnement du serveur. Lorsque elle est étiquetée, elle indique l'échec de l'achèvement de la commande associée. La forme non étiquetée indique un avertissement ; la commande peut encore s'achever avec succès. Le texte lisible par l'homme décrit la condition.

Exemple :

```
C: A222 COPY 1:2 owatagusiam
S: * NO Le disque est plein à 98 %, prière de supprimer les données non nécessaires
S: A222 OK COPY achevé
C: A223 COPY 3:200 blurrybloop
S: * NO Le disque est plein à 98 %, prière de supprimer les données non nécessaires
S: * NO Le disque est plein à 98 %, prière de supprimer les données non nécessaires
S: A223 NO COPY a échoué : le disque est plein
```

7.1.3 Réponse BAD

Contenu : code de réponse FACULTATIF
texte lisible par l'homme

La réponse BAD indique une erreur message de la part du serveur. Lorsque elle est étiquetée, elle rapporte une erreur de niveau protocole dans la commande du client ; l'étiquette indique la commande qui cause l'erreur. La forme non étiquetée

indique une erreur de niveau protocole pour laquelle la commande associée ne peut être déterminée ; elle peut aussi indiquer un échec interne du serveur. Le texte lisible par l'homme décrit la condition.

Exemple :

C: ...Très longue ligne de commande...

S: * BAD Ligne de commande trop longue

C: ...ligne vide...

S: * BAD Ligne de commande vide

C: A443 EXPUNGE

S: * BAD Défaillance du disque, tentative de sauvegarde sur un nouveau disque !

S: * OK Sauvegarde réussie, pas de perte de données

S: A443 OK Expunge achevé

7.1.4 Réponse PREAUTH

Contenu : code de réponse FACULTATIF
texte lisible par l'homme

La réponse PREAUTH est toujours non étiquetée et est un des trois accueils possibles au début de connexion. Elle indique que la connexion a déjà été authentifiée par des moyens externes ; et donc qu'aucune commande LOGIN n'est nécessaire.

Exemple : S: * PREAUTH IMAP4rev1 connexion au serveur sous le nom de Smith

7.1.5 Réponse BYE

Contenu : code de réponse FACULTATIF
texte lisible par l'homme

La réponse BYE est toujours non étiquetée, et indique que le serveur va clore la connexion. Le texte lisible par l'homme PEUT être affiché à l'utilisateur par le client dans un rapport d'état. La réponse BYE est envoyée sous une de quatre conditions :

- 1) Au titre d'une séquence logout normale. Le serveur va clore la connexion après l'envoi de la réponse OK étiquetée à la commande LOGOUT.
- 2) Comme annonce de clôture d'urgence. Le serveur ferme immédiatement la connexion.
- 3) Comme annonce d'une auto clôture sur inactivité. Le serveur ferme immédiatement la connexion.
- 4) Comme un des trois accueils possibles au début de connexion, indiquant que le serveur ne veut pas accepter une connexion avec ce client. Le serveur ferme immédiatement la connexion.

La différence entre un BYE qui survient au titre d'une séquence LOGOUT normale (le premier cas) et un BYE qui survient à cause d'une défaillance (les trois autres cas) est que la connexion ferme immédiatement dans le cas de défaillance. Dans tous les cas, le client DEVRAIT continuer à lire les données de réponse provenant du serveur jusqu'à la fermeture de la connexion ; cela assure que toute réponse en cours non étiquetée ou d'achèvement est lue et traitée.

Exemple : S: * BYE Auto fermeture ; inactivité trop longue

7.2 Réponses du serveur – État du serveur et de la boîte aux lettres

Ces réponses sont toujours non étiquetées. C'est la façon dont les données d'état de serveur et de boîte aux lettres sont transmises du serveur au client. Beaucoup de ces réponses résultent normalement d'une commande du même nom.

7.2.1 Réponse CAPABILITY

Contenu : Liste des capacités

La réponse CAPABILITY survient par suite d'une commande CAPABILITY. La liste des capacités contient une liste des noms de capacités, séparés par une espace, que le serveur prend en charge. La liste des capacités DOIT inclure l'atome "IMAP4rev1".

De plus, les mises en œuvre de client et de serveur DOIVENT mettre en œuvre les capacités STARTTLS, LOGINDISABLED, et AUTH=PLAIN (décrites dans la [RFC2595]). Voir à la section 11 "Considérations pour la sécurité" des informations importantes.

Un nom de capacité qui commence par "AUTH=" indique que le serveur prend en charge ce mécanisme d'authentification particulier.

La capacité LOGINDISABLED indique que la commande LOGIN est désactivée, et que le serveur répondra par une réponse NO étiquetée à toute tentative d'utiliser la commande LOGIN même si le nom et le mot de passe de l'utilisateur sont valides. Un client IMAP NE DOIT PAS produire la commande LOGIN si le serveur annonce la capacité LOGINDISABLED.

D'autres noms de capacité indiquent que le serveur prend en charge une extension, révision, ou amendement du protocole IMAP4rev1. Les réponses de serveur DOIVENT se conformer au présent document jusqu'à ce que le client produise une commande qui utilise la capacité associée.

Les noms de capacité DOIVENT commencer par "X" ou être des extensions, révisions ou amendements standard IMAP4rev1 ou en cours de normalisation enregistrés auprès de l'IANA. Un serveur NE DOIT PAS offrir de noms de capacité non enregistrés ou non standard, si de tels noms ne portent pas le préfixe "X".

Les mises en œuvre de client NE DEVRAIENT PAS exiger de noms de capacités autres que "IMAP4rev1", et DOIVENT ignorer tout nom de capacité inconnu.

Un serveur PEUT envoyer automatiquement des capacités, en utilisant le code de réponse CAPABILITY dans les réponses initiales PREAUTH ou OK, et en envoyant un code de réponse CAPABILITY mis à jour dans la réponse OK étiquetée au titre d'une authentification réussie. Il n'est pas nécessaire qu'un client envoie une commande CAPABILITY séparée si il reconnaît ces capacités automatiques.

Exemple: S: * CAPABILITY IMAP4rev1 STARTTLS AUTH=GSSAPI XPIG-LATIN

7.2.2 Réponse LIST

Contenu : attributs de nom
délimitateur de hiérarchie
nom

La réponse LIST survient par suite d'une commande LIST. Elle retourne un seul nom qui correspond à la spécification de LIST. Il peut y avoir plusieurs réponses LIST pour une seule commande LIST.

Quatre attributs de nom sont définis :

`\Noinferiors`

Il n'est possible à aucun niveau de hiérarchie fils d'exister sous ce nom ; aucun niveau fils n'existe actuellement et aucun ne pourra être créé à l'avenir.

`\Noselect`

Il n'est pas possible d'utiliser ce nom comme boîte aux lettres sélectionnée.

`\Marked`

La boîte aux lettres a été marquée "intéressante" par le serveur ; la boîte aux lettres contient probablement des messages qui ont été ajoutés depuis la dernière fois que la boîte aux lettres a été sélectionnée.

`\Unmarked`

La boîte aux lettres ne contient aucun message supplémentaire depuis la dernière fois qu'elle a été sélectionnée.

Si le serveur ne peut pas déterminer si la boîte aux lettres est "intéressante" ou non, le serveur NE DEVRAIT PAS envoyer `\Marked` ni `\Unmarked`. Le serveur NE DOIT PAS envoyer plus d'un `\Marked`, `\Unmarked`, et `\Noselect` pour une seule boîte aux lettres, et PEUT n'en envoyer aucun.

Le délimiteur hiérarchique est un caractère utilisé pour délimiter les niveaux hiérarchiques dans un nom de boîte aux lettres. Un client peut l'utiliser pour créer des boîtes aux lettres filles, et pour chercher des niveaux supérieurs ou inférieurs de la hiérarchie de dénomination. Tous les enfants d'un nœud hiérarchique supérieur DOIVENT utiliser le même caractère séparateur. Un délimiteur hiérarchique NIL signifie qu'aucune hiérarchie n'existe ; le nom est un nom "plat".

Le nom représente une hiérarchie non ambiguë de gauche à droite, et DOIT être valide pour être utilisé comme référence dans les commandes LIST et LSUB. Sauf si `\Noselect` est indiqué, le nom DOIT aussi être valide comme argument pour les

commandes, telles que SELECT, qui acceptent les noms de boîte aux lettres.

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

7.2.3 Réponse LSUB

Contenu : attributs de nom
délimateur hiérarchique
nom

La réponse LSUB survient par suite d'une commande LSUB. Elle retourne un seul nom qui correspond à la spécification LSUB. Il peut y avoir plusieurs réponses LSUB pour une seule commande LSUB. Le format des données est identique à celui de la réponse LIST.

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

7.2.4 Réponse STATUS

Contenu: nom
liste d'états entre parenthèses

La réponse STATUS survient par suite d'une commande STATUS. Elle retourne le nom de boîte aux lettres qui correspond à la spécification de STATUS et les informations d'état de la boîte aux lettres demandée.

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

7.2.5 Réponse SEARCH

Contenu : zéro, un ou plusieurs numéros

La réponse SEARCH survient par suite d'une commande SEARCH ou UID SEARCH. Le ou les numéros se réfèrent aux messages qui correspondent aux critères de recherche. Pour SEARCH, ce sont les numéros de séquence du message ; pour UID SEARCH, ce sont les identifiants univoques. Chaque numéro est délimité par une espace.

Exemple : S: * SEARCH 2 3 6

7.2.6 Réponse FLAGS

Contenu : liste de fanions entre parenthèses

La réponse FLAGS survient par suite d'une commande SELECT ou EXAMINE. La liste de fanions entre parenthèses identifie les fanions (au minimum, les fanions définis par le système) qui sont applicables à cette boîte aux lettres. Des fanions autres que les fanions système peuvent aussi exister, selon la mise en œuvre de serveur.

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

Exemple : S: * FLAGS (\Répondu \Fanion \Supprimé \Vu \Brouillon)

7.3 Réponses du serveur – Taille de boîte aux lettres

Ces réponses sont toujours non étiquetées. C'est la façon dont les changements de la taille de la boîte aux lettres sont transmis du serveur au client. Immédiatement après le jeton "*" se trouve un numéro qui représente un compte de message.

7.3.1 Réponse EXISTS

Contenu : aucun

La réponse EXISTS rapporte le nombre de messages dans la boîte aux lettres. Cette réponse survient par suite d'une commande SELECT ou EXAMINE, et si la taille de la boîte aux lettres change (par exemple, avec de nouveaux messages).

La mise à jour à partir de la réponse EXISTS DOIT être enregistrée par le client.

Exemple : S: * 23 EXISTS

7.3.2 Réponse RECENT

Contenu : aucun

La réponse RECENT fait rapport du nombre de messages qui ont le fanion \Recent mis. Cette réponse survient par suite d'une commande SELECT ou EXAMINE, et si la taille de la boîte aux lettres change (par exemple, nouveaux messages).

Note : Il n'est pas garanti que les numéros de séquence de message des messages récents soient dans une gamme contiguë des n numéros les plus élevés des messages de la boîte aux lettres (où n est la valeur rapportée par la réponse RECENT). Des exemples de situations dans lesquelles ce n'est pas le cas sont : plusieurs clients qui ont la même boîte aux lettres ouverte (la première session notifiée la verra comme récente, les autres la verront probablement comme non récente), et lorsque la boîte aux lettres est réordonnée par un agent non IMAP.

La seule façon fiable pour identifier des messages récents est de regarder les fanions de message pour voir ceux qui ont le fanion \Recent mis, ou de faire une commande SEARCH RECENT.

La mise à jour à partir de la réponse RECENT DOIT être enregistrée par le client.

Exemple : S: * 5 RECENT

7.4 Réponses du serveur – état du message

Ces réponses sont toujours non étiquetées. C'est la façon dont les données d'un message sont transmises du serveur au client, souvent par suite d'une commande du même nom. Immédiatement après le jeton "*" se trouve un nombre qui représente un numéro de séquence de message.

7.4.1 Réponse EXPUNGE

Contenu : aucun

La réponse EXPUNGE rapporte que le numéro de séquence de message spécifié a été retiré de façon permanente de la boîte aux lettres. Le numéro de séquence du message pour chaque message successif dans la boîte aux lettres est immédiatement décrémenté de 1 et cette diminution est reflétée dans le numéro de séquence de message des réponses suivantes (y compris les autres réponses EXPUNGE non étiquetées).

La réponse EXPUNGE décrémenté aussi le nombre des messages de la boîte aux lettres ; il n'est pas nécessaire d'envoyer une réponse EXISTS avec la nouvelle valeur.

Il résulte de la règle du décrément immédiat que les numéros de séquence de message qui apparaissent dans un ensemble de réponses EXPUNGE successives dépend de ce que les messages sont retirés en commençant des numéros inférieurs vers les supérieurs ou des supérieurs vers les inférieurs. Par exemple, si les cinq derniers messages d'une boîte aux lettres qui en contient neuf sont effacés, un serveur "d'inférieur vers supérieur" va envoyer cinq réponses EXPUNGE non étiquetées pour le numéro de séquence du message 5, alors que un serveur "de supérieur vers inférieur" va envoyer des réponses EXPUNGE non étiquetées successives pour les numéros de séquence de message 9, 8, 7, 6, et 5.

Une réponse EXPUNGE NE DOIT PAS être envoyée lorsque aucune commande n'est en cours, ni en répondant à une commande FETCH, STORE ou SEARCH. Cette règle est nécessaire pour empêcher une perte de synchronisation de numéro de séquence de message entre client et serveur. Une commande n'est pas "en cours" tant que la commande complète n'a pas été reçue ; en particulier, une commande n'est pas "en cours" durant la négociation d'une continuation de commande.

Note : UID FETCH, UID STORE et UID SEARCH sont des commandes différentes de FETCH, STORE et SEARCH. Une réponse EXPUNGE PEUT être envoyée durant une commande UID.

La mise à jour à partir de la réponse EXPUNGE DOIT être enregistrée par le client.

Exemple : S: * 44 EXPUNGE

7.4.2 Réponse FETCH

Contenu : données de message

La réponse FETCH retourne des données sur un message au client. Les données sont des paires de noms d'élément de données et leurs valeurs entre parenthèses. Cette réponse survient par suite d'une commande FETCH ou STORE, ainsi que par des décisions unilatérales du serveur (par exemple, des mises à jour de fanions).

Les éléments de données courants sont :

BODY

C'est une forme de BODYSTRUCTURE sans données d'extension.

BODY[<section>]<<octet-d'origine>>

C'est une chaîne exprimant le contenu du corps de la section spécifiée. La chaîne DEVRAIT être interprétée par le client conformément au codage de transfert de contenu, au type de corps, et au sous-type.

Si l'octet d'origine est spécifié, cette chaîne est une sous-chaîne du contenu de corps entier, commençant à cet octet d'origine. Cela signifie que BODY[<0> PEUT être tronqué, mais BODY[] n'est JAMAIS tronqué.

Note : La facilité d'octet d'origine NE DOIT PAS être utilisée par un serveur dans une réponse FETCH sauf si le client le demande spécifiquement au moyen d'un FETCH d'un élément de données BODY[<section>]<<partial>>.

Les données textuelles à 8 bits sont permises si un identifiant de CHARSET fait partie de la liste de paramètres de corps entre parenthèses de cette section. Noter que les en-têtes (spécificateurs de partie HEADER ou MIME, ou la portion d'en-tête d'une partie MESSAGE/RFC822), DOIVENT être à 7 bits ; les caractères à 8 bits ne sont pas permis dans les en-têtes. Noter aussi que la ligne blanche de délimitation de la [RFC2822] entre l'en-tête et le corps n'est pas affectée par les sous-ensembles de ligne d'en-tête ; la ligne blanche est toujours incluse au titre des données d'en-tête, sauf dans le cas d'un message sans corps et sans ligne blanche.

Les données non textuelles telles que les données binaires DOIVENT avoir le codage de transfert en forme textuelle, tel que le BASE64, avant d'être envoyées au client. Pour déduire les données binaires d'origine, le client DOIT décoder la chaîne codée de transfert.

BODYSTRUCTURE

C'est une liste entre parenthèses qui décrit la structure de corps MIME-IMB d'un message. Elle est calculée par le serveur qui analyse les champs d'en-tête MIME-IMB, en prenant en tant que de besoin les divers champs par défaut.

Par exemple, un simple message de texte de 48 lignes et 2279 octets peut avoir une structure de corps de : ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48)

Les parties multiples sont indiquées en les incorporant entre parenthèses. Au lieu d'un type de corps comme premier élément de la liste entre parenthèses, il y a une séquence d'une ou plusieurs structures de corps incorporées. Le second élément de la liste à parenthèses est le sous-type multipartie (mixte, résumé, parallèle, alternatif, etc.).

Par exemple, un message en deux parties consistant en un texte et une pièce jointe de texte codé en BASE64 peut avoir une structure de corps de : (("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'extension suivent le sous-type multipartie. Les données d'extension ne sont jamais retournées avec la recherche BODY, mais peuvent être retournées avec une recherche BODYSTRUCTURE. Les données d'extension, si elles sont présentes, DOIVENT être dans l'ordre défini. Les données d'extension d'une partie de corps multipartie sont dans l'ordre suivant :

liste entre parenthèses des paramètres de corps

C'est une liste entre parenthèses de paires attribut/valeur [par exemple, ("foo" "bar" "baz" "rag") où "bar" est la valeur de "foo", et "rag" est la valeur de "baz"] telles que définies dans la [RFC2045].

disposition de corps

C'est une liste entre parenthèses, comportant une chaîne de type de disposition, suivie par une liste entre parenthèses des

paires d'attribut/valeur de disposition telles que définies dans la [RFC2183].

langage du corps

C'est une chaîne ou une liste entre parenthèses qui donne la valeur du langage du corps telle que définie dans la [RFC3282].

localisation du corps

C'est une chaîne qui donne l'URI du contenu du corps, tel que défini dans la [RFC2557].

Aucune des données d'extension suivantes n'est encore définie dans la présente version de ce protocole. De telles données d'extension peuvent consister en zéro, un ou plusieurs NIL, chaînes, numéros, ou listes entre parenthèses éventuellement incorporées de telles données. Les mises en œuvre de client qui font une recherche de BODYSTRUCTURE DOIVENT être prêtes à accepter de telles données d'extension. Les mises en œuvre de serveur NE DOIVENT PAS envoyer de telles données d'extension tant qu'elles ne sont pas définies par une révision du présent protocole.

Les champs de base d'une partie de corps non multipartie sont dans l'ordre suivant :

type de corps

Une chaîne donnant le nom du type de support de contenu tel que défini dans la [RFC2045].

sous-type de corps

Une chaîne donnant le nom du sous-type de contenu tel que défini dans la [RFC2045].

liste entre parenthèse des paramètres du corps

Une liste entre parenthèses de paires d'attribut/valeur [par exemple, ("foo" "bar" "baz" "rag") où "bar" est la valeur de "foo" et "rag" est la valeur de "baz"] telle que définie dans la [RFC2045].

identifiant de corps

Une chaîne donnant l'identifiant de contenu tel que défini dans la [RFC2045].

description de corps

Une chaîne donnant la description du contenu telle que défini dans la [RFC2045].

codage de corps

Une chaîne donnant le codage de transfert du contenu tel que défini dans la [RFC2045].

taille de corps

Un nombre donnant la taille du corps en octets. Noter que cette taille est celle du codage de transfert et non la taille résultante après tout 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 lignes de texte du message encapsulé.

Un type de corps de type TEXT contient, immédiatement après les champs de base, la taille du corps en lignes de texte. Noter que cette taille est celle du codage de transfert du contenu et non la taille résultante après décodage.

Les données d'extension suivent les champs de base et les champs spécifiques du type dont la liste figure ci-dessus. Les données d'extension ne sont jamais retournées avec la recherche BODY, mais peuvent être retournées avec une recherche BODYSTRUCTURE. Les données d'extension, si elles sont présentes, DOIVENT être dans l'ordre défini.

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

corps MD5

Une chaîne donnant la valeur du corps MD5 telle que définie dans [MD5].

disposition du corps

Une liste entre parenthèses avec les mêmes contenu et fonction que la disposition de corps d'une partie de corps multipartie.

langage du corps

Une chaîne ou liste entre parenthèses donnant la valeur du langage du corps telle que définie dans la [RFC3282].

localisation du corps

Une chaîne donnant l'URI du contenu du corps tel que défini dans la [RFC2557].

Les données d'extension suivantes ne sont pas encore définies dans la présente version du protocole, et seront comme décrit ci dessus pour les données d'extension multipartie.

ENVELOPE

Une liste entre parenthèses qui décrit la structure de l'enveloppe d'un message. Elle est calculée par le serveur en analysant l'en-tête [RFC2822] dans les parties composantes, en prenant les divers champs par défaut si nécessaire.

Les champs de la structure d'enveloppe sont dans l'ordre suivant : date, sujet, origine, expéditeur, réponse à, à, cc, bcc, en-réponse-à, et message-id. Les champs date, sujet, en-réponse-à, et message-id sont des chaînes. Les champs from, sender, reply-to, to, cc, et bcc sont des listes entre parenthèses de structures d'adresse.

Une structure d'adresse est une liste entre parenthèse qui décrit une adresse de messagerie électronique. Les champs d'une structure d'adresse sont dans l'ordre suivant : nom de la personne, liste de domaine SMTP (route de source), nom de boîte aux lettres, et nom de l'hôte.

La syntaxe de groupe de la [RFC2822] est indiquée par une forme spéciale de la structure d'adresse dans laquelle le champ de nom d'hôte est NIL. Si le champ de nom de boîte aux lettres est aussi NIL, c'est une extrémité de marqueur de groupe (point-virgule dans la syntaxe de la RFC822). Si le champ de nom de boîte aux lettres n'est pas NIL, c'est le début d'un marqueur de groupe, et le champ de nom de boîte aux lettres détient la phrase de nom de groupe.

Si les lignes d'en-tête Date, Subject, In-Reply-To, et Message-ID sont absentes dans l'en-tête de la [RFC2822], le membre correspondant de l'enveloppe est NIL ; si ces lignes d'en-tête sont présentes mais vides, le membre correspondant de l'enveloppe est la chaîne vide.

Note : Certains serveurs peuvent retourner un membre d'enveloppe NIL dans le cas "présent mais vide". Les clients DEVRAIENT traiter NIL et la chaîne vide de façon identique.

Note : La [RFC2822] exige que tous les messages aient un en-tête Date valide. Donc, le membre Date dans l'enveloppe ne peut pas être NIL ou la chaîne vide.

Note : La [RFC2822] exige que les en-têtes In-Reply-To et Message-ID, s'ils sont présents, aient un contenu non vide. Donc, les membres in-reply-to et message-id dans l'enveloppe ne peuvent pas être la chaîne vide.

Si les lignes d'en-tête From, To, cc, et bcc sont absentes dans l'en-tête de la [RFC2822], ou sont présents mais vides, le membre correspondant de l'enveloppe est NIL.

Si les lignes Sender ou Reply-To sont absentes dans l'en-tête de la [RFC2822], ou sont présentes mais vides, le serveur règle le membre correspondant de l'enveloppe comme étant la même valeur que le membre from (le client n'est pas censé savoir faire cela).

Note : La [RFC2822] exige que tous les messages aient un en-tête From valide. Donc, les membres from, sender, et reply-to dans l'enveloppe ne peuvent pas être NIL.

FLAGS

Une liste entre parenthèses de fanions qui sont établis pour ce message.

INTERNALDATE

Une chaîne qui représente la date interne du message.

RFC822

Équivalent à BODY[
].

RFC822.HEADER

Équivalent à BODY[HEADER]. Noter qu'il ne résulte pas de cela que \Vu soit établi parce que les données de réponse RFC822.HEADER surviennent par suite d'une commande FETCH de RFC822.HEADER. Les données de réponse BODY[HEADER] surviennent par suite de FETCH de BODY[HEADER] (qui établit \Vu) ou BODY.PEEK[HEADER] (qui n'établit pas \Vu).

RFC822.SIZE

Un nombre exprimant la taille du message selon la [RFC2822].

RFC822.TEXT

Équivalent de BODY[TEXT].

UID

Un nombre exprimant l'identifiant univoque du message.

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

7.5 Réponses du serveur – Demande de continuation de commande

La réponse de demande de continuation de commande est indiquée par un jeton "+" au lieu d'une étiquette. Cette forme de réponse indique que le serveur est prêt à accepter la continuation d'une commande provenant du client. Le reste de cette réponse est une ligne de texte.

Cette réponse est utilisée dans la commande AUTHENTICATE pour transmettre les données du serveur au client, et demande des données supplémentaires du client. Cette réponse est aussi utilisée si un argument de n'importe quelle commande est littéral.

Il n'est pas permis au client d'envoyer les octets du littéral sauf si le serveur indique qu'il est attendu. Cela permet au serveur de traiter les commandes et de rejeter les erreurs ligne par ligne. Le reste de la commande, y compris le CRLF qui termine une commande, suit les octets du littéral. Si il n'y a pas d'arguments de commande supplémentaires, les octets du littéral sont suivis d'une espace et de ces arguments.

Exemple :

```
C: A001 LOGIN {11}
S: + Prêt pour du texte additionnel de commande
C: FRED FOOBAR {7}
S: + Prêt pour du texte additionnel de commande
C: gros bonhomme
S: A001 OK LOGIN achevé
C: A044 BLURDYBLOOP {102856}
S: A044 BAD Pas de commande telle que "BLURDYBLOOP"
```

8. Exemple de connexion IMAP4rev1

Ce qui suit est une transcription d'une connexion IMAP4rev1. Une longue ligne de cet exemple est coupée pour faciliter la lisibilité.

```
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] Le message 17 est le premier message non lu
S: * OK [UIDVALIDITY 3857529045] Les UID sont valides
S: a002 OK [READ-WRITE] SELECT achevé
C: a003 fetch 12 full
S: * 12 FETCH (FLAGS (\Seen) INTERNALDATE "17-Jul-1996 02:44:25 -0700"
RFC822.SIZE 4286 ENVELOPE ("Wed, 17 Jul 1996 02:23:25 -0700 (PDT)"
"Sommaire et compte-rendu de la réunion du groupe de travail IMAP4rev1
("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" "MIT.EDU")) NIL NIL
"<B27397-0100000@cac.washington.edu>")
BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 302892))
S: a003 OK FETCH achevé
C: a004 fetch 12 body[header]
```

S: * 12 FETCH (BODY[HEADER] {342})
 S: Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
 S: From: Terry Gray <gray@cac.washington.edu>
 S: Subject: Sommaire et compte-rendu de la réunion du groupe de travail IMAP4rev1
 S: To: imap@cac.washington.edu
 S: cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@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 achevé
 C: a005 store 12 +flags \Supprimé
 S: * 12 FETCH (FLAGS (\Lu \Supprimé))
 S: a005 OK +FLAGS achevé
 C: a006 logout
 S: * BYE IMAP4rev1 Le serveur termine la connexion
 S: a006 OK LOGOUT achevé

9. Syntaxe formelle

La spécification de syntaxe suivante utilise la notation de forme Backus-Naur augmentée (ABNF, *Augmented Backus-Naur Form*) telle que spécifiée dans [ABNF].

Dans le cas de règles facultatives ou de remplacement dans lequel une règle plus récente recoupe une règle plus ancienne, la règle plus ancienne DOIT avoir la priorité. Par exemple, lorsque "\Lu" est analysé comme fanion et a le nom de fanion \Lu et non d'une extension de fanion, il peut être analysé comme une extension de fanion. Certaines instances de cette règle sont notées ci-dessous, mais pas toutes.

Note : Les règles de l'ABNF DOIVENT être strictement suivies ; en particulier :

- (1) Excepté lorsque noté autrement, tous les caractères alphabétiques sont insensibles à la casse. L'utilisation de caractères majuscules ou minuscules pour définir les noms des jetons n'est que pour faciliter la lecture. Les mises en œuvre DOIVENT accepter ces chaînes de façon insensible à la casse.
- (2) Dans tous les cas, SP se réfère à exactement une espace. Il N'EST PAS permis d'y substituer TAB, d'insérer des espaces additionnelles, ou de traiter SP autrement que comme équivalent à LWSP.
- (3) Le caractère ASCII NUL, %x00, NE DOIT PAS être utilisé.

address = "(" addr-name SP addr-adl SP addr-mailbox SP addr-host ")"

addr-adl = nstring
 ; détient le chemin de route-addr de la [RFC2822] sinon NIL

addr-host = nstring
 ; NIL indique la syntaxe de groupe de la [RFC2822].
 ; Autrement, contient le nom de domaine de la [RFC2822].

addr-mailbox = nstring
 ; NIL indique la fin du groupe [RFC2822] ; si non NIL et si addr-host est NIL,
 ; contient le nom de groupe de la [RFC2822].
 ; Autrement, contient la partie locale de la [RFC2822] après suppression des guillemets [RFC2822].

addr-name = nstring
 ; Sinon NIL, détient la phrase provenant de la boîte aux lettres [RFC2822] après suppression des guillemets

append = "APPEND" SP boîte-aux-lettres [SP liste-de-fanions] [SP date-heure] SP littéral

astring = 1*ASTRING-CHAR / chaîne

ASTRING-CHAR = ATOM-CHAR / resp-specials

atome = 1*ATOM-CHAR

ATOM-CHAR = <tout CHAR excepté atom-specials>
 atom-specials = "(" / ")" / "{" / SP / CTL / list-wildcards / quoted-specials / resp-specials
 authenticate = "AUTHENTICATE" SP auth-type *(CRLF base64)
 auth-type = atome ; Défini par [SASL]
 base64 = *(4base64-char) [base64-terminal]
 base64-char = ALPHA / DIGIT / "+" / "/" ; Sensible à la casse
 base64-terminal = (2base64-char "=") / (3base64-char "=")
 body = "(" (body-type-1part / body-type-mpart) ")"
 body-extension = nstring / nombre / "(" body-extension *(SP body-extension) ")"
 ; Future expansion. Les mises en œuvre de client DOIVENT accepter les champs body-extension.
 ; Les mises en œuvre de serveur NE DOIVENT PAS générer de champs body-extension sauf comme
 ; défini par de futures révisions standard ou en cours de normalisation de cette spécification.
 body-ext-1part = body-fld-md5 [SP body-fld-dsp [SP body-fld-lang [SP body-fld-loc *(SP body-extension)]]]
 ; NE DOIT PAS être retourné sur une recherche "BODY" non extensible.
 body-ext-mpart = body-fld-param [SP body-fld-dsp [SP body-fld-lang [SP body-fld-loc *(SP body-extension)]]]
 ; NE DOIT PAS être retourné sur une recherche "BODY" non extensible.
 body-fields = body-fld-param SP body-fld-id SP body-fld-desc SP body-fld-enc SP body-fld-octets
 body-fld-desc = nstring
 body-fld-dsp = "(" chaîne SP body-fld-param ")" / nil
 body-fld-enc chaîne = (DQUOTE ("7BIT" / "8BIT" / "BINARY" / "BASE64" / "QUOTED-PRINTABLE") DQUOTE) /
 chaîne
 body-fld-id = nstring
 body-fld-lang = nstring / "(" chaîne *(SP chaîne) ")"
 body-fld-loc = nstring
 body-fld-lines = nombre
 body-fld-md5 = nstring
 body-fld-octets = nombre
 body-fld-param = "(" chaîne SP chaîne *(SP chaîne SP chaîne) ")" / nil
 body-type-1part = (body-type-basic / body-type-msg / body-type-text) [SP body-ext-1part]
 body-type-basic = media-basic SP body-fields
 ; Le sous-type MESSAGE subtype NE DOIT PAS être "RFC822"
 body-type-mpart = 1*body SP media-subtype [SP body-ext-mpart]
 body-type-msg = media-message SP body-fields SP envelope SP body SP body-fld-lines
 body-type-text = media-text SP body-fields SP body-fld-lines
 capability = ("AUTH=" auth-type) / atome

; Les nouvelles capacités DOIVENT commencer par "X" ou être enregistrées auprès de l' IANA
; comme normalisées ou en cours de normalisation.

capability-data "CAPABILITY" *(SP capability) SP "IMAP4rev1" *(SP capability)
; Les serveurs DOIVENT mettre en œuvre les capacités STARTTLS, AUTH=PLAIN, et LOGINDISABLED
; Les serveurs qui offrent la compatibilité avec la RFC 1730 DOIVENT lister "IMAP4" comme première capacité.

CHAR8 = %x01-ff ; Tout OCTET sauf NUL, %x00

charset = atome / guillemeté

command = tag SP (command-any / command-auth / command-nonauth / command-select) CRLF
; Modal fondé sur l'état.

command-any = "CAPABILITY" / "LOGOUT" / "NOOP" / x-command
; Valide dans tous les états.

command-auth = append / create / delete / examine / list / lsub / rename / select / status / subscribe / unsubscribe
; Valide seulement dans l'état Authentifié ou Sélectionné

command-nonauth = login / authenticate / "STARTTLS"
; Valide seulement dans l'état Non authentifié

command-select = "CHECK" / "CLOSE" / "EXPUNGE" / copy / fetch / store / uid / search
; Valide seulement dans l'état Sélectionné.

continue-req = "+" SP (resp-text / base64) CRLF

copy = "COPY" SP ensemble-de-séquence SP boîte-aux-lettres

create = "CREATE" SP boîte-aux-lettres ; L'utilisation de INBOX donne une erreur NO.

date = date-text / DQUOTE date-text DQUOTE

date-day = 1*2DIGIT ; Jours du mois

date-day-fixed = (SP DIGIT) / 2DIGIT ; Version à format fixe de la date du jour.

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 = DQUOTE date-day-fixed "-" date-month "-" date-year SP time SP zone DQUOTE

delete = "DELETE" SP boîte-aux-lettres ; L'utilisation de INBOX donne une erreur NO.

digit-nz = %x31-39 ; 1 à 9

envelope = "(" env-date SP env-subject SP env-from SP env-sender SP env-reply-to SP env-to SP env-cc SP env-bcc SP env-in-reply-to SP 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-sende = "(" 1*address ")" / nil
 env-subject = nstring
 env-to = "(" 1*address ")" / nil
 examine = "EXAMINE" SP boîte-aux-lettres
 fetch = "FETCH" SP ensemble-de-séquence SP ("ALL" / "FULL" / "FAST" / fetch-att / "(" fetch-att *(SP fetch-att) ")")
 fetch-att = "ENVELOPE" / "FLAGS" / "INTERNALDATE" / "RFC822" [".HEADER" / ".SIZE" / ".TEXT"] / "BODY" ["STRUCTURE"] / "UID" / "BODY" section ["<" nombre "." nz-nombre ">"] / "BODY.PEEK" section ["<" nombre "." nz-nombre ">"]
 flag = "\Répondu" / "\Fanion" / "\Supprimé" / "\Lu" / "\Brouillon" / flag-keyword / flag-extension
 ; N'inclut pas "\Récent".
 flag-extension = "\" atome
 ; Future expansion. Les mises en œuvre de client DOIVENT accepter les fanions d'extension de fanion.
 ; Les mises en œuvre de serveur NE DOIVENT PAS générer de fanions d'extension de fanion, sauf
 ; comme défini par de futures révisions standard ou en cours de normalisation de cette spécification.
 flag-fetch = flag / "\Récent"
 flag-keyword = atome
 flag-list = "(" [flag *(SP flag)] ")"
 flag-perm = flag / "\"
 greeting = "*" SP (resp-cond-auth / resp-cond-bye) CRLF
 header-fld-name = astring
 header-list = "(" header-fld-name *(SP header-fld-name) ")"
 list = "LIST" SP boîte-aux-lettres SP liste-de-boîtes-aux-lettres
 liste-de-boîtes-aux-lettres = 1*list-char / chaîne
 list-char = ATOM-CHAR / list-wildcards / resp-specials
 list-wildcards = "%" / "*"
 littéral = "{" nombre "}" CRLF *CHAR8
 ; Nombre représente le nombre de caractères.
 login = "LOGIN" SP userid SP password
 lsub = "LSUB" SP boîte-aux-lettres SP liste-de-boîtes-aux-lettres
 boîte-aux-lettres = "INBOX" / astring
 ; INBOX est insensible à la casse. Toutes les variantes de INBOX (par exemple, "iNbOx") DOIVENT être
 ; interprétées comme INBOX et non comme astring. Une astring qui comporte la séquence insensible à la casse
 ; "I" "N" "B" "O" "X" est considérée comme INBOX et non une astring. Voir au paragraphe 5.1 les détails
 ; sémantiques des noms de boîte aux lettres.
 boîte-aux-lettres-data = "FLAGS" SP flag-list / "LIST" SP liste-de-boîte-aux-lettres /

"LSUB" SP liste-de-boîte-aux-lettres / "SEARCH" *(SP nz-nombre) /
 "STATUS" SP boîte-aux-lettres SP "(" [status-att-list] ")" / nombre SP "EXISTS" / nombre SP "RECENT"

liste-de-boîte-aux-lettres = "(" [mbx-list-flags] ")" SP (DQUOTE QUOTED-CHAR DQUOTE / nil) SP boîte-aux-lettres

mbx-list-flags = *(mbx-list-oflag SP) mbx-list-sflag *(SP mbx-list-oflag) / mbx-list-oflag *(SP mbx-list-oflag)

mbx-list-oflag = "\Noinferiors" / flag-extension
 ; Autres fanions ; plusieurs possible par réponse LIST.

mbx-list-sflag = "\Noselect" / "\Marked" / "\Unmarked"
 ; Fanions indiquant la possibilité d'un choix ; un seul par réponse LIST.

media-basic = ((DQUOTE ("APPLICATION" / "AUDIO" / "IMAGE" / "MESSAGE" / "VIDEO") DQUOTE) /
 chaîne) SP media-subtype
 ; Défini dans la [RFC2046]

media-message = DQUOTE "MESSAGE" DQUOTE SP DQUOTE "RFC822" DQUOTE
 ; Défini dans la [RFC2046]

media-subtype = chaîne
 ; Défini dans la [RFC2046]

media-text = DQUOTE "TEXT" DQUOTE SP media-subtype
 ; Défini dans la [RFC2046]

message-data = nz-nombre SP ("EXPUNGE" / ("FETCH" SP msg-att))

msg-att = "(" (msg-att-dynamic / msg-att-static) *(SP (msg-att-dynamic / msg-att-static)) ")"

msg-att-dynamic = "FLAGS" SP "(" [flag-fetch *(SP flag-fetch)] ")"
 ; PEUT changer pour un message.

msg-att-static = "ENVELOPE" SP envelope / "INTERNALDATE" SP date-time / "RFC822" [".HEADER" /
 ".TEXT"] SP nstring / "RFC822.SIZE" SP nombre / "BODY" ["STRUCTURE"] SP body /
 "BODY" section ["<" nombre ">"] SP nstring / "UID" SP uniqueid
 ; NE DOIT PAS changer pour un message.

nil = "NIL"

nstring = chaîne / nil

nombre = 1*DIGIT ; Entier de 32 bits non signé ($0 \leq n < 4\ 294\ 967\ 296$).

nz-nombre = digit-nz *DIGIT ; Entier positif de 32 bits non signé ($0 < n < 4\ 294\ 967\ 296$).

password = astring

guillemeté = DQUOTE *QUOTED-CHAR DQUOTE

QUOTED-CHAR = <tout TEXT-CHAR sauf quoted-specials> / "\" quoted-specials

quoted-specials = DQUOTE / "\"

rename = "RENAME" SP boîte-aux-lettres SP boîte-aux-lettres
 ; L'utilisation de INBOX comme destination donne une erreur NO.

response = *(continue-req / response-data) response-done

response-data = "*" SP (resp-cond-state / resp-cond-bye / boîte-aux-lettres-data / message-data / capability-data) CRLF

response-done = response-tagged / response-fatal

response-fatal = "*" SP resp-cond-by CRLF ; Le serveur ferme immédiatement la connexion.

response-tagged = tag SP resp-cond-state CRLF

resp-cond-auth = ("OK" / "PREAUTH") SP resp-text
 ; Condition d'authentification.

resp-cond-by = "BYE" SP resp-text

resp-cond-state = ("OK" / "NO" / "BAD") SP resp-text
 ; Condition d'état.

resp-specials = "]"

resp-text = "[" resp-text-code "]" SP text

resp-text-code = "ALERT" / "BADCHARSET" [SP "(" charset *(SP charset) ")"] / capability-data / "PARSE" /
 "PERMANENTFLAGS" SP "(" [flag-perm *(SP flag-perm)] ")" / "READ-ONLY" /
 "READ-WRITE" / "TRYCREATE" / "UIDNEXT" SP nz-nombre / "UIDVALIDITY" SP nz-nombre /
 "UNSEEN" SP nz-nombre / atome [SP 1*<tout TEXT-CHAR excepté "]">]

search = "SEARCH" [SP "CHARSET" SP charset] 1*(SP search-key)
 ; L'argument CHARSET DOIT être enregistré auprès de l'IANA.

search-key = "ALL" / "ANSWERED" / "BCC" SP astring / "BEFORE" SP date / "BODY" SP astring /
 "CC" SP astring / "DELETED" / "FLAGGED" / "FROM" SP astring /
 "KEYWORD" SP flag-keyword / "NEW" / "OLD" / "ON" SP date / "RECENT" / "SEEN" /
 "SINCE" SP date / "SUBJECT" SP astring / "TEXT" SP astring / "TO" SP astring /
 "UNANSWERED" / "UNDELETED" / "UNFLAGGED" / "UNKEYWORD" SP flag-keyword /
 "UNSEEN" /
 ; Au-dessus de cette ligne étaient en IMAP2 "DRAFT" / "HEADER" SP header-fld-name SP astring /
 "LARGER" SP nombre / "NOT" SP search-key / "OR" SP search-key SP search-key /
 "SENTBEFORE" SP date / "SENTON" SP date / "SENTSINCE" SP date / "SMALLER" SP nombre /
 "UID" SP ensemble-de-séquence / "UNDRAFT" / ensemble-de-séquence /
 "(" search-key *(SP search-key) ")"

section = "[" [section-spec] "]"

section-msgtext = "HEADER" / "HEADER.FIELDS" [".NOT"] SP header-list / "TEXT"
 ; niveau supérieur ou partie de MESSAGE/RFC822.

section-part = nz-nombre *("." nz-nombre)
 ; incorporation de la partie de corps.

section-spec = section-msgtext / (section-part ["." section-text])

section-text = section-msgtext / "MIME"
 ; texte autre que la partie de corps réel (en-têtes, etc.)

select = "SELECT" SP boîte-aux-lettres

seq-nombre = nz-nombre / "*"

 ; numéro de séquence du message (commandes COPY, FETCH, STORE) ou identifiant univoque (commandes

 ; UID COPY, UID FETCH, UID STORE). * représente le plus grand numéro utilisé. Dans le cas de numéros de

 ; séquence du message, c'est le nombre de messages dans une boîte aux lettres non vide. Dans le cas d'identifiant

 ; univoque, c'est l'identifiant univoque du dernier message de la boîte aux lettres ou, si la boîte aux lettres est vide,

 ; c'est la valeur UIDNEXT actuelle de la boîte aux lettres. Le serveur devrait répondre par un BAD étiqueté à une

 ; commande qui utilise un numéro de séquence de message supérieur au nombre des messages de la boîte aux

 ; lettres choisie. Cela inclut "*" si la boîte aux lettres choisie est vide.

seq-range = seq-nombre ":" seq-nombre

 ; deux valeurs de seq-nombre et toutes les valeurs entre elles sans considération de l'ordre.

 ; Exemple : 2:4 et 4:2 sont équivalentes et indiquent les valeurs 2, 3, et 4.

; Exemple : une gamme de séquence d'identifiants univoques de 3291:* inclut l'UID du dernier message de la boîte aux lettres, même si cette valeur est inférieure à 3291.

ensemble-de-séquence = (seq-nombre / seq-range) ["," ensemble-de-séquence]
 ; ensemble de valeurs de seq-nombre, sans considération de l'ordre. Les serveurs PEUVENT fusionner les chevauchements et/ou exécuter la séquence dans n'importe quel ordre. Exemple : un ensemble de numéros de séquence de message de 2,4:7,9,12:* pour une boîte aux lettres avec 15 messages est équivalent à 2,4,5,6,7,9,12,13,14,15. Exemple : un ensemble de numéros de séquence de message de *:4,5:7 pour une boîte aux lettres de 10 messages est équivalent à 10,9,8,7,6,5,4,5,6,7 et PEUT être réordonné et la fusion des chevauchements être 4,5,6,7,8,9,10.

status = "STATUS" SP boîte-aux-lettres SP "(" status-att *(SP status-att) ")"

status-att = "MESSAGES" / "RECENT" / "UIDNEXT" / "UIDVALIDITY" / "UNSEEN"

status-att-val = ("MESSAGES" SP nombre) / ("RECENT" SP nombre) / ("UIDNEXT" SP nz-nombre) / ("UIDVALIDITY" SP nz-nombre) / ("UNSEEN" SP nombre)

status-att-val = ("MESSAGES" SP nombre) / ("RECENT" SP nombre) / ("UIDNEXT" SP nz-nombre) / ("UIDVALIDITY" SP nz-nombre) / ("UNSEEN" SP nombre)

status-att-list = status-att-val *(SP status-att-val)

store = "STORE" SP ensemble-de-séquence SP store-att-flags

store-att-flags = (["+" / "-"] "FLAGS" [".SILENT"]) SP (flag-list / (flag *(SP flag)))

chaîne = guillemeté / littéral

subscribe = "SUBSCRIBE" SP boîte-aux-lettres

tag = 1* <tout ASTRING-CHAR excepté "+">

text = 1*TEXT-CHAR

TEXT-CHAR = <tout CHAR sauf CR et LF>

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

uid = "UID" SP (copy / fetch / search / store)
 ; Identifiants uniques utilisés à la place des numéros de séquence de messages.

uniqueid = nz-nombre ; Strictement ascendant.

unsubscribe = "UNSUBSCRIBE" SP boîte aux lettres

userid = astring

x-command = "X" atome <arguments de commande expérimentale>

zone = ("+" / "-") 4DIGIT
 ; Valeur de quatre chiffres signée de hhmm représentant les heures et minutes à l'Est de Greenwich (c'est à dire, la valeur dont l'heure donnée diffère du Temps Universel). Soustraire le décalage horaire de l'heure donnée ; va donner la forme UT. La zone de Temps Universel est "+0000".

10. Note de l'auteur

Le présent document est une révision ou réécriture de documents plus anciens, et subroge les spécifications de protocole de ces documents : RFC 2060, RFC 1730, le document IMAP2bis.TXT non publié, RFC 1176, et RFC 1064.

11. Considérations pour la sécurité

Les transactions du protocole IMAP4rev1, y compris les données de messagerie électronique, sont envoyées en clair sur le réseau sauf négociation de moyens de protection contre l'espionnage. Cela peut être réalisé par l'utilisation de STARTTLS, par la protection négociée de confidentialité dans la commande AUTHENTICATE, ou par quelque autre mécanisme de protection.

11.1 Considérations pour la sécurité sur STARTTLS

La spécification de la commande STARTTLS et de la capacité LOGINDISABLED dans ce document remplace celle de la [RFC2595]. La [RFC2595] reste normative pour l'authentifiant PLAIN de SASL.

Les mises en œuvre de client et serveurs IMAP DOIVENT mettre en œuvre la suite de chiffrement TLS_RSA_WITH_RC4_128_MD5 [RFC2246], et DEVRAIENT mettre en œuvre la suite de chiffrement TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA [RFC2246]. Ceci est important car cela assure que deux mises en œuvre conformes quelconques peuvent être configurées de façon à interopérer. Toutes les autres suites de chiffrement sont FACULTATIVES. Noter que cela constitue un changement par rapport au paragraphe 2.1 de la [RFC2595].

Durant la négociation TLS, le client DOIT vérifier sa compréhension du nom d'hôte du serveur par rapport à l'identité du serveur telle que présentée dans le message Certificate du serveur, afin d'empêcher les attaques par interposition. Si la vérification échoue, le client DEVRAIT soit demander une confirmation explicite de l'usager, soit terminer la connexion et indiquer que l'identité du serveur est suspecte. La vérification est effectuée conformément aux règles suivantes :

Le client DOIT utiliser le nom d'hôte du serveur dont il s'est servi pour ouvrir la connexion comme valeur à comparer au nom du serveur tel qu'exprimé dans le certificat du serveur. Le client NE DOIT PAS utiliser de forme dérivée du nom de serveur à partir d'une source distante non sûre (par exemple, une recherche DNS non sécurisée). La canonisation du CNAME n'est pas effectuée.

Si une extension subjectAltName du type dNSName est présente dans le certificat, elle DEVRAIT être utilisée comme source de l'identité du serveur.

La vérification de correspondance est insensible à la casse.

Un caractère générique "*" PEUT être utilisé comme composant de nom le plus à gauche dans le certificat. Par exemple, *.example.com correspondrait à a.example.com, à foo.example.com, etc. mais ne correspondrait pas à example.com.

Si le certificat contient plusieurs noms (par exemple, plus d'un champ dNSName), une correspondance avec l'un de ses champs est considérée comme acceptable.

Client et serveur DOIVENT tous deux vérifier le résultat de la commande STARTTLS et de la négociation TLS ultérieure pour voir si une authentification ou confidentialité acceptable a été réalisée.

11.2 Autres considérations pour la sécurité

Un message d'erreur du serveur pour une commande AUTHENTICATE qui échoue à cause d'accréditifs invalides NE DEVRAIT PAS préciser pourquoi les accréditifs sont invalides.

L'utilisation de la commande LOGIN envoie les mots de passe en clair. Cela peut être évité par l'utilisation de la commande AUTHENTICATE avec un mécanisme SASL qui n'utilise pas de mots de passe en clair, en négociant d'abord le chiffrement via STARTTLS ou quelque autre mécanisme de protection.

Une mise en œuvre de serveur DOIT mettre en œuvre une configuration qui, au moment de l'authentification, exige :

- (1) Que la commande STARTTLS ait été négociée. OU
- (2) Que d'autres mécanismes de protection de la session contre la divulgation de mots de passe soient fournis. OU
- (3) Que les mesures suivantes soient en place :
 - (a) La capacité LOGINDISABLED est annoncée, et des mécanismes SASL (tels que PLAIN) utilisant des mots de passe en clair NE SONT PAS annoncés dans la liste des capacités. ET
 - (b) La commande LOGIN retourne une erreur même si le mot de passe est correct. ET
 - (c) La commande AUTHENTICATE retourne une erreur pour tous les mécanismes SASL qui utilisent des mots

de passe en clair, même si le mot de passe est correct.

Un message d'erreur du serveur pour une commande LOGIN défaillante NE DEVRAIT PAS spécifier que le nom d'utilisateur, par opposition au mot de passe, est invalide.

Un serveur DEVRAIT avoir des mécanismes pour limiter ou retarder des tentatives de AUTHENTICATE/LOGIN qui échouent.

Des considérations supplémentaires sur la sécurité sont exposées dans la section sur les commandes AUTHENTICATE et LOGIN.

12. Considérations relatives à l'IANA

Les capacités IMAP4 sont enregistrées lors de la publication de RFC en cours de normalisation ou expérimentales approuvées par l'IESG. Le registre est actuellement situé à :

<http://www.iana.org/assignments/imap4-capabilities>

Comme la présente spécification révisé les extensions STARTTLS et LOGINDISABLED précédemment définies dans la [RFC2595], le registre sera mis à jour en conséquence.

Les noms de service GSSAPI/Kerberos/SASL sont enregistrés lors de la publication de RFC en cours de normalisation ou de RFC expérimentales approuvées par l'IESG. Le registre est actuellement situé à :

<http://www.iana.org/assignments/gssapi-service-names>

Comme la présente spécification définit le nom de service "imap" précédemment défini dans la RFC 1731, le registre sera mis à jour en conséquence.

Appendice A Références normatives

Les documents suivants contiennent les définitions ou spécifications qui sont nécessaires pour comprendre correctement le présent document :

[MD5] J. Myers, M. Rose, "Champ d'en-tête Contenu-MD5", octobre 1995, RFC 1864.

[ABNF] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", RFC 2234, novembre 1997.

[RFC2045] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 1 : Format des corps de message Internet", novembre 1996.

[RFC2046] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 2 : Types de support", novembre 1996.

[RFC2047] K. Moore, "MIME (Extensions de messagerie Internet multi-objets) Partie trois : extensions d'en-tête de message pour texte non ASCII", novembre 1996. (*MàJ par [RFC2184](#), [RFC2231](#)*).

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[UTF-7] D. Goldsmith, M. Davis, "UTF-7, un format de transformation de Unicode sûr pour la messagerie", RFC 2152, mai 1997. (*Info.*).

[RFC2183] R. Troost, S. Dorner, K. Moore, éd., "Communication des informations de présentation dans les messages Internet : le champ d'en-tête Contenu-disposition", août 1997. (*MàJ par [RFC2184](#), [RFC2231](#)*).

[RFC2222] J. Myers, "Authentification simple et couche de sécurité (SASL)", octobre 1997. (*Obsolète, voir [RFC4422](#), [RFC4752](#)*) (*MàJ par [RFC2444](#)*) (*Proposition de norme*).

[RFC2245] C. Newman, "Mécanisme SASL anonyme", , novembre 1997. (*Obsolète, voir [RFC4505](#)*).

[RFC2246] T. Dierks et C. Allen, "Protocole TLS version 1.0", janvier 1999.

[RFC2557] J. Palme, A. Hopmann et N. Shelness, "Encapsulation MIME de documents agrégés, tels que HTML (MHTML)", mars 1999,.

[RFC2595] C. Newman, "Utilisation de TLS avec IMAP, POP3 et ACAP", juin 1999. (*MàJ par [RFC4616](#)*).

[RFC2822] P. Resnick, "Format de message Internet", avril 2001.

[RFC2831] P. Leach et C. Newman, "Utilisation de l'authentification par résumé comme mécanisme SASL", mai 2000.

[CHARSET] N. Freed et J. Postel, "Procédures d'enregistrement des jeux de caractère par l'IANA", RFC 2978, BCP 19, octobre 2000.

[RFC3282] H. Alvestrand, "En-têtes de langue du contenu", mai 2002.

Les documents suivants décrivent les questions de qualité de mise en œuvre qui devraient être étudiées avec soin lors de la mise en œuvre du présent protocole :

[IMAP-IMPL] B. Leiba, "Recommandations pour la mise en œuvre de IMAP4", RFC 2683, septembre 1999. (*Information*).

[IMAP-MULTI] M. Gahrns, "Pratique de boîtes aux lettres à accès multiple dans IMAP4", RFC 2180., juillet 1997. (*Information*)

A.1 Références informatives

Les documents suivants décrivent les protocoles concernés :

[IMAP-DISC] R. Austein, "Opérations de synchronisation pour clients IMAP4 déconnectés", Non publiée.

[IMAP-MODEL] M. Crispin, "Modèles de messagerie électronique répartie dans IMAP4", RFC 1733, décembre 1994.

[ACAP] C. Newman, J. G. Myers, "ACAP – Protocole d'accès à la configuration d'application", RFC 2244, novembre 1997.

[SMTP] J. Klensin, éditeur, "Protocole simple de transfert de messagerie", avril 2001, STD 10, RFC 2821.

Les documents suivants sont historiques ou décrivent des aspects historiques de ce protocole :

[RFC2061] M. Crispin, "Compatibilité IMAP4 avec IMAP2bis", décembre 1996. (*Remplace [RFC1730](#) (Information)*)

[RFC1732] M. Crispin, "Compatibilité IMAP4 avec IMAP2 et IMAP2bis", décembre 1994. (*Information*)

[RFC2062] M. Crispin, "Protocole d'accès au message Internet – syntaxe obsolète", décembre 1996. (*Information*).

[RFC1176] M. Crispin, "Protocole d'accès à la messagerie interactive", , août 1990. (*Expérimentale*).

[RFC-822] D. Crocker, "Norme pour le format des messages de texte de l'ARPA-Internet", STD 11, août 1982.

[RFC-821] J. Postel, "Protocole simple de transfert de messagerie", août 1982.

Appendice B Changements par rapport à la RFC 2060

- 1) Précise la description des identifiants univoques et leur sémantique.
- 2) Description corrigée de SELECT pour préciser l'exigence de UIDVALIDITY dans les réponses SELECT et EXAMINE.
- 3) Ajout d'un exemple d'échec de recherche.
- 4) Corrige store-att-flags : "#flag" devrait être "1#flag".
- 5) Précise les règles de recherche et de section.
- 6) Corrige l'exemple STORE.
- 7) Corrige une faute de frappe sur "BASE645".

- 8) Retire les parenthèses inutiles dans l'exemple de message en deux parties avec une pièce jointe de texte et BASE64.
- 9) Retire la réponse "MAILBOX" obsolète de données-de-boîte aux lettres.
- 10) Un "<" parasite dans la règle pour données-de-boîte-aux-lettres a été retiré.
- 11) Ajout d'un CRLF à continue-req.
- 12) Exclusion spécifique de "]" de l'atome dans resp-text-code.
- 13) Précise que clients et serveurs devraient adhérer strictement à la syntaxe du protocole.
- 14) Souligne au 5.2 que EXISTS ne peut pas être utilisé pour réduire une boîte aux lettres.
- 15) Ajout de NEWNAME à resp-text-code.
- 16) Précise que la chaîne vide, et non NIL, est utilisée comme argument pour LIST.
- 17) Précise que NIL peut être retourné comme délimiteur hiérarchique pour la chaîne vide d'argument de nom de boîte aux lettres si l'espace de nom de boîte aux lettres est plat.
- 18) Précise que addr-boîte-aux-lettres et addr-name ont les guillemets de la RFC-2822 retirés.
- 19) Mise à jour de la référence à UTF-7.
- 20) Correction de l'exemple du § 6.3.11.
- 21) Précise que les UID non existants sont ignorés.
- 22) Mise à jour de la référence à la RFC 2183.
- 23) Développement des diagrammes d'état.
- 24) Précise que les réponses de recherche partielle ne sont retournées que dans les réponses à la commande de recherche partielle.
- 25) Ajout du code de réponse UIDNEXT. Correction de la référence à la définition de UIDVALIDITY.
- 26) Précision sur "peut" contre "PEUT".
- 27) Référence à la RFC-2119.
- 28) Précise que les glissements superflus ne sont pas permis en UTF-7 modifié.
- 29) Précise qu'il n'y a pas de glissement implicite en UTF-7 modifié.
- 30) Précise que "INBOX" dans un nom de boîte aux lettres est toujours INBOX, même si il est donné comme une chaîne.
- 31) Ajout de la parenthèse ouverte manquante dans la règle de grammaire de media-basic.
- 32) Corrige la syntaxe d'attribut dans données-de-boîte-aux-lettres.
- 33) Ajout de UIDNEXT aux réponses EXAMINE.
- 34) Précise les réponses UNSEEN, PERMANENTFLAGS, UIDVALIDITY, et UIDNEXT dans SELECT et EXAMINE. Elles sont maintenant exigées mais ne l'étaient pas dans les plus anciennes versions.
- 35) Mise à jour des références avec les numéros des RFC.
- 36) Purge de text-mime2.
- 37) Précise que les noms UTF-7 modifiés doivent être insensibles à la casse et que le viol de cette convention devrait être évité.
- 38) Corrige l'exemple UID FETCH.
- 39) Précise UID FETCH, UID STORE, et UID SEARCH par rapport aux réponses EXPUNGE non étiquetées.
- 40) Précise l'utilisation du mot "convention".
- 41) Précise qu'une commande n'est pas "en cours" tant qu'elle n'a pas été reçue en entier (précisément, qu'une commande n'est pas "en cours" durant la négociation de continuation de commande).
- 42) Précision sur l'enveloppe par défaut.
- 43) Précise que SP signifie un caractère espace et un seul.
- 44) Interdiction des états stupides dans la réponse LIST.
- 45) Précise que ENVELOPE, INTERNALDATE, RFC822*, BODY*, et UID est statique pour un message.
- 46) Ajout du code de réponse BADCHARSET.
- 47) Mise à jour de la syntaxe formelle selon les conventions de l'ABNF.
- 48) Précise le délimiteur hiérarchique de queue dans la sémantique de CREATE.
- 49) Précise que la "ligne blanche" est celle de délimitation de la [RFC-2822].
- 50) Précise que RENAME devrait aussi créer la hiérarchie nécessaire pour l'achèvement de la commande.
- 51) Correction de body-ext-mpart de façon à ne pas exiger de langage si la disposition est présente.
- 52) Précise la réponse RFC822.HEADER.
- 53) Corrige l'espace manquante après la chaîne de jeu de caractère dans une recherche.
- 54) Corrige les guillemets manquants pour BADCHARSET dans resp-text-code.
- 55) Précise que ALL, FAST, et FULL empêchent l'apparition de tout autre élément de données.
- 56) Précise la sémantique d'argument de référence dans LIST.
- 57) Précise qu'une chaîne nulle pour SEARCH HEADER X-FOO signifie tout message avec une ligne d'en-tête ayant un nom de champ de X-FOO sans considération du texte de l'en-tête.
- 58) Mise en réserve spécifique des noms de boîte aux lettres à 8 bits pour utilisation future comme UTF-8.
- 59) Ce n'est pas une erreur du client de mémoriser un fanion qui n'est pas dans la liste PERMANENTFLAGS ; cependant, le serveur ignorera le changement ou ne fera le changement que dans la session.
- 60) Corrige/précise le texte concernant les glissements superflus.
- 61) Corrige les erreurs typographiques dans la section "Changements".
- 62) Précise que STATUS ne doit pas être utilisé pour vérifier les nouveaux messages dans la boîte aux lettres choisie.
- 63) Précise le comportement LSUB avec le caractère générique "%".

- 64) Change AUTHORIZATION en AUTHENTICATE au paragraphe 7.5.
- 65) Précise la description du type de corps multipartite.
- 66) Précise que STORE FLAGS n'affecte pas \Récent.
- 67) Change "Ouest" en "Est" dans la description de timezone.
- 68) Précise que les commandes qui cassent le traitement en parallèle de commandes doivent attendre une réponse d'achèvement.
- 69) Précise que EXAMINE n'affecte pas \Récent.
- 70) Rend cohérente la description de la structure MIME.
- 71) Précise que les recherches de date ne tiennent pas compte de l'heure et du décalage horaire de l'en-tête INTERNALDATE ou Date:. En d'autres termes, "ON 13-APR-2000" signifie les messages avec un texte INTERNALDATE commençant par "13-APR-2000", même si le différentiel de décalage horaire par rapport à la zone horaire est suffisant pour passer cette INTERNALDATE dans le jour précédent ou suivant.
- 72) Précise que la recherche d'en-tête n'ajoute pas de ligne blanche si il y en a déjà une dans le message [RFC-2822].
- 73) Précise (dans l'exposé sur les UID) que les messages sont immuables
- 74) Ajoute un exemple de recherche de CHARSET.
- 75) Précise dans SEARCH que les mots clés sont un type de fanion.
- 76) Précise la nature obligatoire des réponses de données SELECT.
- 77) Ajout du code de réponse CAPABILITY facultatif dans le OK ou PREAUTH initial.
- 78) Ajout d'une note disant que le serveur peut envoyer une commande CAPABILITY non étiquetée au titre des réponses à AUTHENTICATE et LOGIN.
- 79) Suppression de la déclaration sur l'inutilité de produire une commande CAPABILITY plus d'une fois dans une connexion. Cette déclaration n'est plus vraie.
- 80) Précise que la commande EXPUNGE non étiquetée décrémente le nombre de messages de la boîte aux lettres.
- 81) Corrige la définition de "corps" (l'enchaînement a de plus forts liens que l'entrelacement).
- 82) Ajout d'un nouveau paragraphe "Notes spéciales pour les mises en œuvre" se référant à [IMAP-IMPL].
- 83) Précise que la réponse CAPABILITY non étiquetée à une commande AUTHENTICATE ne devrait être faite que s'il n'a pas été négocié de couche de sécurité.
- 84) Changement de la définition de atome pour exclure "]". Mise à jour de astring pour inclure "]" pour la rétro compatibilité. Suppression de resp-text-atom.
- 85) Suppression de NEWNAME. Cela ne peut pas fonctionner parce que les noms de boîte aux lettres peuvent être littéraux et peuvent inclure "]". Cette fonctionnalité peut être traitée via l'arbitrage.
- 86) Déplacement des raisons de l'UTF-7 modifié afin d'avoir un enchaînement plus logique des paragraphes.
- 87) Précision de la garantie d'unicité de l'UID par l'utilisation de DOIT.
- 88) Note que les clients devraient lire les données de réponse jusqu'à la clôture de la connexion au lieu de la clore immédiatement avec un BYE.
- 89) Changement des références de la RFC-822 en celles de la RFC-2822.
- 90) Précision que la RFC-2822 devrait être suivie plutôt que la RFC-822.
- 91) Changement de la recommandation de capacités facultatives automatiques dans LOGIN et AUTHENTICATE pour l'utilisation du code de réponse CAPABILITY dans le OK étiqueté. Cela favorise plus l'interopérabilité qu'une réponse CAPABILITY non étiquetée non sollicitée.
- 92) STARTTLS et AUTH=PLAIN sont de mise en œuvre obligatoire ; ajout de recommandations pour les autres mécanismes [SASL].
- 93) Précision qu'une "connexion" (par opposition à un "serveur" ou une "commande") est dans un de quatre états.
- 94) Précise qu'une commande échouée ou rejetée ne change pas l'état.
- 95) Partage les références en normatives et informatives.
- 96) Discute les questions d'échec d'authentification dans une section sur la sécurité.
- 97) Précise qu'un élément de données n'est pas nécessairement d'un seul type de données.
- 98) Précise que les gammes de séquence sont indépendantes de l'ordre.
- 99) Changement d'un exemple pour préciser que les glissements superflus en UTF-7 modifié ne peuvent être réglés en omettant simplement le glissement. La chaîne toute entière doit être recalculée.
- 100) Changement de la définition de Structure d'enveloppe dans la mesure où la [RFC-2822] utilise "enveloppe" pour se référer à l'enveloppe [SMTP] et non aux données d'enveloppe qui apparaissent dans l'en-tête de la [RFC-2822].
- 101) Expansion des données de réponse RFC822.HEADER par opposition à BODY[HEADER].
- 102) Précision de la sémantique de l'état Logout, changement de l'article ASCII.
- 103) Changements de la sécurité pour se conformer aux exigences de l'IESG.
- 104) Ajout de la définition d'URI de corps.
- 105) Coupure de la définition de gamme de séquence selon trois règles, avec des descriptions révisées pour chacune.
- 106) Importation de STARTTLS et LOGINDISABLED de la [RFC2595].
- 107) Ajout de la section Considérations relatives à l'IANA.
- 108) Précise les hypothèses valides du client sur les UID de nouveau message par rapport à UIDNEXT.
- 109) Précise que les changements aux fanions permanents affectent les sessions concurrentes aussi bien sue successives.
- 110) Précise que l'état authentifié peut être introduit pas la commande CLOSE.
- 111) Souligne que SELECT et EXAMINE sont les exceptions à la règle qu'un échec d'une commande ne change pas l'état.

- 112) Précise que les messages qui viennent d'être ajoutés ont le fanion Récent mis.
 113) Précise que les messages nouvellement copiés DEVRAIENT avoir le fanion Récent établi.
 114) Précise que les commandes UID retournent toujours l'UID dans les réponses FETCH.
 115) Ajout de la prise en charge de la localisation de contenu (Content-Location) dans BODYSTRUCTURE.

Appendice C. Index des mots clés

+FLAGS <liste-de-fanions> (mémoire l'élément de données de commande)	34
+FLAGS.SILENT <liste-de-fanions> (mémoire l'élément de données de commande)	34
-FLAGS <liste-de-fanions> (mémoire l'élément de données de commande)	34
-FLAGS.SILENT <liste-de-fanions> (mémoire l'élément de données de commande)	34
ALERT (code de réponse)	37
ALL (élément de recherche)	29
ALL (clé de recherche)	31
ANSWERED (clé de recherche)	31
APPEND (commande)	26
AUTHENTICATE (commande)	16
BAD (réponse)	38
BADCHARSET (code de réponse)	37
BCC <chaîne> (clé de recherche)	29
BEFORE <date> (clé de recherche)	29
BODY (élément de recherche)	29
BODY (résultat de recherche)	43
BODY <chaîne> (clé de recherche)	29
BODY.PEEK[<section>]<<partiel>> (élément de recherche)	33
BODYSTRUCTURE (élément de recherche)	33
BODYSTRUCTURE (résultat de recherche)	43
BODY[<section>]<<octet-d'origine>> (résultat de recherche)	43
BODY[<section>]<<partiel>> (élément de recherche)	32
BYE (réponse)	39
CAPABILITY (commande)	13
CAPABILITY (code de réponse)	37
CAPABILITY (réponse)	39
CC <chaîne> (clé de recherche)	29
CHECK (commande)	27
CLOSE (commande)	28
COPY (commande)	34
CREATE (commande)	20
Date interne (attribut de message)	6
DELETE (commande)	20
DELETED (clé de recherche)	29
DRAFT (clé de recherche)	29
ENVELOPE (élément de recherche)	33
ENVELOPE (résultat de recherche)	45
EXAMINE (commande)	19
EXISTS (réponse)	42
EXPUNGE (commande)	28
EXPUNGE (réponse)	42
Fanions (attribut de message)	6
Fanion de session (classe de fanion)	6
Fanion permanent (classe de fanion)	6
Fanion système (type de fanion)	6
FAST (élément de recherche)	32
FETCH (commande)	31
FETCH (réponse)	45
FLAGGED (clé de recherche)	29
FLAGS (élément de recherche)	33
FLAGS (résultat de recherche)	45
FLAGS (réponse)	41
FLAGS <liste-de-fanions> (mémoire l'élément de données de commande)	34
FLAGS.SILENT <liste-de-fanions> (mémoire l'élément de données de commande)	34

FROM <chaîne> (clé de recherche)	29
FULL (élément de recherche)	32
HEADER (spécificateur de partie)	32
HEADER <field-name> <chaîne> (clé de recherche)	29
HEADER.FIELDS <header-list> (spécificateur de partie)	32
HEADER.FIELDS.NOT <header-list> (spécificateur de partie)	32
Identifiant univoque (UID) (attribut de message)	5
INTERNALDATE (élément de recherche)	33
INTERNALDATE (résultat de recherche)	45
KEYWORD <flag> (clé de recherche)	29
LARGER <n> (clé de recherche)	29
LIST (commande)	23
LIST (réponse)	40
LOGIN (commande)	17
LOGOUT (commande)	15
LSUB (commande)	25
LSUB (réponse)	41
MESSAGES (élément d'état)	26
MIME (spécificateur de partie)	32
Mot clé (type de fanion)	6
NEW (clé de recherche)	29
NIL	10
NO (réponse)	38
NOOP (commande)	14
NOT <search-key> (clé de recherche)	30
Numéro de séquence de message (attribut de message)	6
OK (réponse)	38
OLD (clé de recherche)	30
ON <date> (clé de recherche)	30
OR <search-key1> <search-key2> (clé de recherche)	30
PARSE (code de réponse)	37
PERMANENTFLAGS (code de réponse)	37
PREAUTH (réponse)	39
READ-ONLY (code de réponse)	38
READ-WRITE (code de réponse)	38
RECENT (réponse)	42
RECENT (clé de recherche)	31
RECENT (élément d'état)	26
RENAME (commande)	21
RFC822 (élément de recherche)	33
RFC822 (résultat de recherche)	46
RFC822.HEADER (élément de recherche)	33
RFC822.HEADER (résultat de recherche)	46
RFC822.SIZE (élément de recherche)	33
RFC822.SIZE (résultat de recherche)	46
RFC822.TEXT (élément de recherche)	34
RFC822.TEXT (résultat de recherche)	46
SEARCH (commande)	29
SEARCH (réponse)	41
SEEN (clé de recherche)	31
SELECT (commande)	18
SENTBEFORE <date> (clé de recherche)	31
SENTON <date> (clé de recherche)	31
SENTSINCE <date> (clé de recherche)	31
SINCE <date> (clé de recherche)	31
SMALLER <n> (clé de recherche)	31
STARTTLS (commande)	15
STATUS (commande)	25
STATUS (réponse)	41
STORE (commande)	34
Structure d'enveloppe (attribut de message)	7
Structure de corps (attribut de message)	7
SUBJECT <chaîne> (clé de recherche)	31

SUBSCRIBE (commande)	22
Taille [RFC-2822] (attribut de message)	7
TEXT (spécificateur de partie)	32
TEXT <chaîne> (clé de recherche)	31
TO <chaîne> (clé de recherche)	31
TRYCREATE (code de réponse)	38
UID (commande)	35
UID (élément de recherche)	34
UID (résultat de recherche)	46
UID <ensemble de séquence> (clé de recherche)	31
UIDNEXT (code de réponse)	38
UIDNEXT (élément d'état)	26
UIDVALIDITY (code de réponse)	38
UIDVALIDITY (élément d'état)	26
UNANSWERED (clé de recherche)	31
UNDELETED (clé de recherche)	31
UNDRAFT (clé de recherche)	31
UNFLAGGED (clé de recherche)	31
UNKEYWORD <flag> (clé de recherche)	31
UNSEEN (code de réponse)	38
UNSEEN (clé de recherche)	31
UNSEEN (élément d'état)	26
UNSUBSCRIBE (commande)	23
X<atome> (commande)	36
\Répondu (fanion système)	6
\Supprimé (fanion système)	6
\Brouillon (fanion système)	6
\Fanion (fanion système)	6
\Marked (attribut de nom de boîte aux lettres)	40
\Noinferiors (attribut de nom de boîte aux lettres)	40
\Noselect (attribut de nom de boîte aux lettres)	41
\Récent (fanion système)	6
\Lu (fanion système)	6
\Unmarked (attribut de nom de boîte aux lettres)	41

Adresse de l'auteur

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

Déclaration de copyright

Copyright (C) The Internet Society (2006).

Le présent document est soumis aux droits, licences et restrictions contenus dans le BCP 78, et à www.rfc-editor.org, et sauf pour ce qui est mentionné ci-après, les auteurs conservent tous leurs droits.

Le présent document et les informations y contenues sont fournies sur une base "EN L'ÉTAT" et Le contributeur, l'organisation qu'il ou elle représente ou qui le/la finance (s'il en est), la INTERNET SOCIETY et la INTERNET ENGINEERING TASK FORCE déclinent toutes garanties, exprimées ou implicites, y compris mais non limitées à toute garantie que l'utilisation des informations ci encloses ne violent aucun droit ou aucune garantie implicite de commercialisation ou d'aptitude à un objet particulier.

Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourrait être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la

mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à <http://www.ietf.org/ipr>.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org.

Remerciement

Le financement de la fonction d'édition des RFC est actuellement fourni par l'activité de soutien administratif (IASA) de l'IETF.