Groupe de travail Réseau
Request pour Comments : 4551
RFC mise à jour : 3501
Catégorie : En cours de normalisation

A. Melnikov, Isode Ltd.
S. Hole, ACI WorldWide/MessagingDirect
juin 2006
Traduction Claude Brière de L'Isle

Extension à IMAP pour l'opération STORE conditionnelle ou la resynchronisation des changements rapides de fanion

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 reporter à l'édition en cours des "normes officielles du protocole Internet" (STD 1) pour connaître l'état de la normalisation et le statut du présent protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Souvent, plusieurs clients IMAP [RFC3501] ont besoin de coordonner les changements à une boîte aux lettres IMAP commune. Comme exemple de cette situation, il y a le cas de clients différents qui travaillent au nom du même utilisateur, et celui de plusieurs utilisateurs qui accèdent à des boîtes aux lettres partagées. Ces clients ont besoin d'un mécanisme pour synchroniser les changements d'état pour les messages contenus dans la boîte aux lettres. Il doivent être capables de garantir qu'un seul client peut changer l'état d'un message (par exemple, les fanions de message) à un instant donné. Un exemple d'une telle application est l'utilisation d'une boîte aux lettres IMAP comme file d'attente de messages avec plusieurs clients en sortie de la file d'attente.

La facilité de mémorisation conditionnelle fournit un mécanisme de mise à jour protégée pour les informations d'état de message qui peuvent détecter et résoudre des conflits entre plusieurs clients qui écrivent des messages.
La facilité de mémorisation conditionnelle permet aussi à un client de resynchroniser rapidement les changements de fanion de messagerie.

Le présent document définit une extension à IMAP [RFC3501].

Table des matières

1. Introduction et généralités

L'extension STORE conditionnelle est présente dans toute mise en œuvre IMAP4 qui retourne "CONDSTORE" comme une des capacités prises en charge dans la réponse à la commande CAPABILITY.

Un serveur IMAP qui prend en charge cette extension DOIT associer une valeur positive non signée de 64 bits appelée une séquence de modification (mod-sequence) à tout message IMAP. C'est une valeur opaque mise à jour par le serveur chaque fois qu'un élément de métadonnées est modifié. Le serveur DOIT garantir que chaque commande STORE effectuée sur la même boîte aux lettres (incluant des mémorisations simultanées sur des éléments de métadonnées différents provenant de différentes connexions) va obtenir une valeur differente de mod-sequence. Aussi, pour tout couple d'opérations STORE réussies effectuées dans la même session sur la même boîte aux lettres, la mod-sequence de la seconde opération achevée DOIT être supérieure à la mod-sequence de la première. Noter que cette dernière règle interdit l'utilisation de l'horloge système comme mod-sequence, parce que si l'heure système change (par exemple, un client NTP [RFC1305] qui ajuste l'heure) la prochaine valeur générée pourrait être inférieure à celle de la précédente.

Les séquences de modification permettent à un client qui prend en charge l'extension CONDSTORE de déterminer si les métadonnées d'un message ont changé depuis un certain moment connu. Chaque fois que l'état d'un fanion change (c'est-à-dire, le fanion est ajouté là où il n'était pas établi, ou le fanion est retiré alors qu'il était établi auparavant) la valeur de la séquence de modification pour le message DOIT être mise à jour. L'ajout du fanion lorsque il est déjà présent ou son retrait quand il n'est pas présent NE DEVRAIT PAS changer la mod-sequence.

Quand un message est ajouté à une boîte aux lettres (via la commande IMAP APPEND, COPY à la boîte aux lettres, ou en utilisant un mécanisme externe) le serveur génère une nouvelle séquence de modification qui est supérieure à la plus forte séquence de modification de tous les messages dans la boîte aux lettres et l'alloue au message ajouté.

Le serveur PEUT mémoriser des valeurs de séquence de modification séparées (par message) pour des éléments de métadonnées différents. Si le serveur fait ainsi, la séquence de modification par message est la plus forte séquence de modification de tous les éléments de métadonnées pour le message spécifié.

Le serveur qui prend en charge cette extension n'est pas obligé d'être capable de mémoriser les séquences de modification pour toutes les boîtes aux lettres disponibles. Le paragraphe 3.1.2 décrit comment le serveur peut agir si une boîte aux lettres particulière ne prend pas en charge la mémorisation persistante des séquences de modification.

Cette extension fait les changements suivants au protocole IMAP4 :
a) ajoute le modificateur UNCHANGEDSINCE STORE,
b) ajoute le code de réponse MODIFIED qui devrait être utilisé avec une réponse OK à la commande STORE. (Il peut aussi être utilisé dans une réponse NO.)
c) ajoute un nouvel élément de données de message MODSEQ à utilser avec la commande FETCH,
d) ajoute le modificateur CHANGEDSINCE FETCH,
e) ajoute un nouveau critère de recherche MODSEQ,
f) étend la syntaxe des réponses SEARCH non étiquetées pour inclure mod-sequence,
g) ajoute de nouvelles réponses OK non étiquetées aux commandes SELECT et EXAMINE,
h) définit un paramètre supplémentaire aux commandes SELECT/EXAMINE,
I) ajoute l'élément de données d'état HIGHESTMODSEQ à la commande STATUS.

Un client qui prend en charge l'estension CONDSTORE indique sa volonté de recevoir des mises à jour de mod-sequence dans toutes les réponses FETCH non étiquetées en produisant :
- une commande SELECT ou EXAMINE avec le paramètre CONDSTORE,
- une commande STATUS (HIGHESTMODSEQ) ,
- une commande FETCH ou SEARCH qui inclut l'élément de données de message MODSEQ,
- une commande FETCH avec le modificateur CHANGEDSINCE, ou
- une commande STORE avec le modificateur UNCHANGEDSINCE.

Le serveur DOIT inclure les données de mod-sequence dans toutes les réponses FETCH non étiquetées suivantes (jusqu'à ce que la connexion soit close) qu'elles aient été causées par un STORE régulier, un STORE avec un modificateur UNCHANGEDSINCE, ou par un agent externe.

Le présent document utilise le terme "client à capacité CONDSTORE" pour se référer à un client qui annonce sa volonté de recevoir des mises à jour de mod-sequence comme décrit ci-dessus. Le terme "commande d'activation de CONDSTORE" va se référer à toute commande figurant sur la liste ci-dessus. Une future extension au présent document pourra étendre la

liste des commandes d'activation de CONDSTORE. Une première commande d'activation de CONDSTORE exécutée dans la session DOIT causer le retour par le serveur de HIGHESTMODSEQ (paragraphe 3.1.1) sauf si le serveur a envoyé le code de réponse NOMODSEQ (paragraphe 3.1.2) quand la boîte aux lettres actuelle a été choisie.

Le reste du document décrit les changements du protocole plus en détails.

2. Conventions utilisées dans le présent document

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

Dans les exemples, les lignes qui commencent par un "S:" sont envoyées par le serveur IMAP, et les lignes qui commencent par "C:" sont envoyées par le client. Des coupures de ligne peuvent apparaître dans les exemples de commandes pour des raisons de clarté de lecture ; quand elles sont présentes dans le message , elles sont représentées par "CRLF".

La syntaxe formelle est définie en utilisant l'ABNF [RFC4234].

Le terme "métadonnées" ou "élément de métadonnées" est utilisé dans ce document. Il se réfère à tout mot clé défini par le système ou l'utilisateur. De futurs documents peuvent étendre "métadonnées" pour inclure d'autres données dynamiques de message.

Certaines boîtes aux lettres IMAP sont privées, accessibles seulement à l'utilisateur qui les possède. D'autres boîtes aux lettres ne le sont pas, soit parce que le possesseur a établi une liste de contrôle d'accès [RFC4314] qui permet l'accès à d'autres utilisateurs, ou parce que c'est une boîte aux lettres partagée. Disons qu'un élément de métadonnées est "partagé" pour la boîte aux lettres si tout changement des éléments de métadonnées est persistant et visible à tous les autres utilisateurs qui accèdent à la boîte aux lettres. Autrement, l'élément de métadonnées est appelé "privé". Noter que les éléments de métadonnées privés sont quand même visibles par toutes les sessions qui accèdent à la boîte aux lettres par le même utilisateur. Noter aussi que des boîtes aux lettres différentes peut avoir des éléments de métadonnées différents comme étant partagés.

Voir à la Section 1 la définition de "client à capacité CONDSTORE" et de "commande d'activation de CONDSTORE".

3. Changements au protocole IMAP

3.1 Nouvelles réponses OK non étiquetées pour SELECT et EXAMINE

Le présent document ajoute deux nouveaux codes de réponse, HIGHESTMODSEQ et NOMODSEQ. Un de ces codes de réponse DOIT être retourné dans la réponse OK non étiquetée pour une commande SELECT/EXAMINE réussie.

À l'ouverture d'une boîte aux lettres, le serveur doit vérifier si la boîte aux lettres prend en charge la mémorisation persistante des séquences de modification. Si la boîte aux lettres prend en charge la mémorisation persistante des mod-sequences et si l'opération d'ouverture réussit, le serveur DOIT envoyer la réponse OK non étiquetée incluant le code de réponse HIGHESTMODSEQ. Si la mémorisation persistante n'est pas prise en charge pour la boîte aux lettres, le serveur DOIT envoyer la réponse OK non étiquetée, incluant à la place le code de réponse NOMODSEQ.

3.1.1 Code de réponse HIGHESTMODSEQ

Le présent document ajoute un nouveau code de réponse qui est retourné dans la réponse OK non étiquetée pour les commandes SELECT et EXAMINE. Un serveur qui prend en charge la mémorisation persistante des mod-sequences pour la boîte aux lettres DOIT envoyer la réponse OK non étiquetée incluant le code de réponse HIGHESTMODSEQ avec chaque commande SELECT ou EXAMINE réussie:

OK [HIGHESTMODSEQ <valeur de mod-sequence>]

où <valeur de mod-sequence> est la plus grande valeur de mod-sequence de tous les messages de la boîte aux lettres. Quand le serveur change UIDVALIDITY pour une boîte aux lettres, il n'a pas à garder le même HIGHESTMODSEQ pour la boîte aux lettres.

Un client déconnecté peut utiliser la valeur de HIGHESTMODSEQ pour vérifier si il doit rafraîchir les métadonnées provenant du serveur. Si la valeur UIDVALIDITY a changé pour la boîte aux lettres choisie, le client DOIT supprimer la valeur de HIGHESTMODSEQ en antémémoire. Si UIDVALIDITY est le même pour la boîte aux lettres, et si la valeur de HIGHESTMODSEQ mémorisée dans l'antémémoire du client est inférieure à la valeur retournée par le serveur, des éléments de métadonnées ont alors changé sur le serveur depuis la dernière synchronisation, et le client doit mettre à jour son antémémoire. Le client PEUT utiliser SEARCH MODSEQ (paragraphe 3.4) pour trouver exactement quels élements de métadonnées ont changé. Autrement, le client PEUT produire une commande FETCH avec le modificateur CHANGEDSINCE (paragraphe 3.3.1) afin d'aller chercher les données pour tous les messages qui ont des éléments de métadonnées qui ont changé depuis une séquence de modification connue.

Exemple 1 :

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 valides
S: * OK [UIDNEXT 4392] Prochain UID prévu
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen *)] Limited
S: * OK [HIGHESTMODSEQ 715194045007]
S: A142 OK [READ-WRITE] SELECT achevé

3.1.2 Code de réponse NOMODSEQ

Un serveur qui ne prend pas en charge la mémorisation persistante des mod-sequences pour la boîte aux lettres DOIT envoyer la réponse OK non étiquetée incluant le code de réponse NOMODSEQ avec chaque commande SELECT ou EXAMINE réussie. Un serveur qui a retourné le code de réponse NOMODSEQ pour une boîte aux lettres, et qui reçoit ensuite une des commandes suivantes alors que la boîte aux lettres est choisie :

- une commande FETCH avec le modificateur CHANGEDSINCE,
- une commande FETCH ou SEARCH qui inclut l'élément de données de message MODSEQ, ou
- une commande STORE avec le modificateur UNCHANGEDSINCE

DOIT rejeter une telle commande avec la réponse étiquetée BAD.

Exemple 2 :

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 valides
S: * OK [UIDNEXT 4392] Prochain UID prévu
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen *)] Limited
S: * OK [NOMODSEQ] Désolé, le format de cette boîte aux lettres n'accepte pas les modsequences
S: A142 OK [READ-WRITE] SELECT achevé

3.2 Commandes STORE et UID STORE

Le présent document définit le modificateur STORE suivant (voir au paragraphe 2.5 de la [RFC4466]):

UNCHANGEDSINCE <mod-sequence>

Pour chaque message spécifié dans l'ensemble de messages, le serveur effectue ce qui suit. Si la mod-sequence de tout élément de métadonnées du message est égale ou inférieure à la valeur de UNCHANGEDSINCE spécifiée, alors l'opération demandée (comme décrite par l'élément de données du message) est effectuée. Si l'opération réussit, le serveur

DOIT mettre à jour l'attribut mod-sequence du message. Une réponse FETCH non étiquetée DOIT être envoyée, même si le suffixe .SILENT est spécifié, et la réponse DOIT inclure l'élément de données de message MODSEQ. Ceci est exigé pour mettre à jour l'antémémoire du client avec les valeurs correctes de mod-sequence. Voir les détails au paragraphe 3.3.2.

Cependant, si la mod-sequence de tout élément de métadonnées du message est supérieure à la valeur spécifiée de UNCHANGEDSINCE, alors l'opération demandée NE DOIT PAS être effectuée. Dans ce cas, l'attribut mod-sequence du message n'est pas mis à jour, et le numéro de message (ou identifiant univoque dans le cas d'une commande UID STORE) est ajouté à la liste des messages qui ont échoué à l'essai UNCHANGESINCE.

Quand le serveur a fini d'effectuer l'opération sur tous les messages de l'ensemble de messages, il vérifie que la liste des messages qui ont échoué à l'essai UNCHANGESINCE n'est pas vide. Si cette liste n'est pas vide, le serveur DOIT retourner dans la réponse étiquetée un code de réponse MODIFIED. Le code de réponse MODIFIED inclut l'ensemble de messages (pour STORE) ou ensemble d'UID (pour UID STORE) de tous les messages qui ont échoué à l'essai UNCHANGESINCE.

Exemple 3 :

Tous les messages réussissent l'essai UNCHANGESINCE.

C: a103 UID STORE 6,4,8 (UNCHANGEDSINCE 12121230045) +FLAGS.SILENT (\Deleted)
S: * 1 FETCH (UID 4 MODSEQ (12121231000))
S: * 2 FETCH (UID 6 MODSEQ (12121230852))
S: * 4 FETCH (UID 8 MODSEQ (12121130956))
S: a103 OK Conditional Store completed

Exemple 4 :

C: a104 STORE * (UNCHANGEDSINCE 12121230045) +FLAGS.SILENT (\Deleted $Processed)
S: * 50 FETCH (MODSEQ (12111230047))
S: a104 OK Store (conditional) completed

Exemple 5 :

C: c101 STORE 1 (UNCHANGEDSINCE 12121230045) -FLAGS.SILENT (\Deleted)
S: * OK [HIGHESTMODSEQ 12111230047]
S: * 50 FETCH (MODSEQ (12111230048))
S: c101 OK Store (conditional) completed

Le code de réponse HIGHESTMODSEQ a été envoyé par le serveur vraisemblablement parce que c'était la première commande d'activation de CONDSTORE.

Exemple 6 :

En dépit de l'échec de l'opération conditionnelle STORE pour le message 7, le serveur continue de traiter le STORE conditionnel afin de trouver tous les messages qui échouent à l'essai.

C: d105 STORE 7,5,9 (UNCHANGEDSINCE 320162338) +FLAGS.SILENT (\Deleted)
S: * 5 FETCH (MODSEQ (320162350))
S: d105 OK [MODIFIED 7,9] Conditional STORE failed

Exemple 7 :

Le même que ci-dessus, mais le serveur suit la recommandation DEVRAIT du paragraphe 6.4.6 de la [RFC3501].

C: d105 STORE 7,5,9 (UNCHANGEDSINCE 320162338) +FLAGS.SILENT (\Deleted)
S: * 7 FETCH (MODSEQ (320162342) FLAGS (\Seen \Deleted))
S: * 5 FETCH (MODSEQ (320162350))
S: * 9 FETCH (MODSEQ (320162349) FLAGS (\Answered))
S: d105 OK [MODIFIED 7,9] Conditional STORE failed

L'utilisation de UNCHANGEDSINCE avec une séquence de modification de 0 échoue toujours si l'élément de métadonnées existe. Un fanion système DOIT toujours être considéré comme existant, qu'il soit établi ou non.

Exemple 8 :

C: a102 STORE 12 (UNCHANGEDSINCE 0) +FLAGS.SILENT ($MDNSent)
S: a102 OK [MODIFIED 12] Conditional STORE failed

Le client a vérifié la présence du mot clé $MDNSent défini par l'utilisateur.

Note : un client qui essaye de faire un changement atomique à l'état d'un élément de métadonnées (ou ensemble d'éléments de métadonnées) particulier devrait être prêt à traiter le cas où le serveur retourne le code de réponse MODIFIED si l'état de l'élément de métadonnées examiné n'a pas changé (mais l'état d'un autre élément de métadonnées a changé). Ceci est nécessaire, parce que certains serveurs ne mémorisent pas les mod-sequences séparés pour différents éléments de métadonnées. Cependant, une mise en œuvre de serveur DEVRAIT éviter de générer des réponses MODIFIED parasites pour les opérations +FLAGS/-FLAGS STORE, même quand le serveur mémorise une seule mod-sequence par message. La Section 5 décrit comment cela peut être fait.

Sauf si le serveur a inclus un FETCH non sollicité pour metttre à jour la connaissance du client sur les messages qui ont échoué à l'essai UNCHANGEDSINCE, à réception du code de réponse MODIFIED, le client DEVRAIT essayer de voir si les éléments de métadonnées requis ont bien changé en produisant la commande FETCH ou NOOP. Il est RECOMMANDÉ que le serveur évite que le client fasse cela en envoyant une réponse FETCH non sollicitée (exemples 9 et 10).

Si les éléments de métadonnées requis n'ont pas changé, le client DEVRAIT réessayer la commande avec la nouvelle mod-sequence. Le client DEVRAIT permettre un nombre configurable mais raisonnable d'essais (au moins 2).

Exemple 9 :

Dans l'exemple ci-dessous, le serveur retourne le code de réponse MODIFIED sans envoyer d'informations décrivant pourquoi l'opération STORE UNCHANGEDSINCE a échoué.

C: a106 STORE 100:150 (UNCHANGEDSINCE 212030000000) +FLAGS.SILENT ($Processed)
S: * 100 FETCH (MODSEQ (303181230852))
S: * 102 FETCH (MODSEQ (303181230852))
...
S: * 150 FETCH (MODSEQ (303181230852))
S: a106 OK [MODIFIED 101] Conditional STORE failed

Le fanion $Processed a été établi sur le message 101...

C: a107 NOOP
S: * 101 FETCH (MODSEQ (303011130956) FLAGS ($Processed))
S: a107 OK

Ou le fanion n'a pas changé, mais un autre a changé (noter que ce comportement de serveur est déconseillé. Les mises en œuvre de serveur devraient aussi voir à la Section 5)...

C: b107 NOOP
S: * 101 FETCH (MODSEQ (303011130956) FLAGS (\Deleted \Answered))
S: b107 OK

...et le client ressaye l'opération pour le message 101 avec la valeur UNCHANGEDSINCE mise à jour

C: b108 STORE 101 (UNCHANGEDSINCE 303011130956) +FLAGS.SILENT ($Processed)
S: * 101 FETCH (MODSEQ (303181230852))
S: b108 OK Conditional Store completed

Exemple 10 :

Même chose que ci-dessus, mais le serveur évite d'avoir besoin que le client interroge sur les changements.

Le fanion $Processed a été établi sur le message 101 par un autre client...

C: a106 STORE 100:150 (UNCHANGEDSINCE 212030000000) +FLAGS.SILENT ($Processed)
S: * 100 FETCH (MODSEQ (303181230852))
S: * 101 FETCH (MODSEQ (303011130956) FLAGS ($Processed))
S: * 102 FETCH (MODSEQ (303181230852))
...
S: * 150 FETCH (MODSEQ (303181230852))
S: a106 OK [MODIFIED 101] Conditional STORE failed

Ou le fanion n'a pas changé, mais un autre a changé (noter que ce comportement de serveur est déconseillé. Les mises en œuvre de serveur devraient aussi voir la Section 5)...

C: a106 STORE 100:150 (UNCHANGEDSINCE 212030000000) +FLAGS.SILENT ($Processed)
S: * 100 FETCH (MODSEQ (303181230852))
S: * 101 FETCH (MODSEQ (303011130956) FLAGS (\Deleted \Answered))
S: * 102 FETCH (MODSEQ (303181230852))
...
S: * 150 FETCH (MODSEQ (303181230852))
S: a106 OK [MODIFIED 101] Conditional STORE failed

...et le client réessaye l'opération pour le message 101 avec la valeur UNCHANGEDSINCE mise à jour.

C: b108 STORE 101 (UNCHANGEDSINCE 303011130956) +FLAGS.SILENT ($Processed)
S: * 101 FETCH (MODSEQ (303181230852))
S: b108 OK Conditional Store completed

Ou le fanion n'a pas changé, mais un autre a changé (bon comportement de serveur. Les mises en œuvre de serveur devraient aussi voir la Section 5)...

C: a106 STORE 100:150 (UNCHANGEDSINCE 212030000000) +FLAGS.SILENT ($Processed)
S: * 100 FETCH (MODSEQ (303181230852))
S: * 101 FETCH (MODSEQ (303011130956) FLAGS ($Processed \Deleted \Answered))
S: * 102 FETCH (MODSEQ (303181230852))
...
S: * 150 FETCH (MODSEQ (303181230852))
S: a106 OK Conditional STORE completed

Exemple 11 :

L'exemple suivant se fonde sur celui du paragraphe 4.2.3 de la [RFC2180] et montre que le code de réponse MODIFIED peut être aussi retourné dans la réponse NO étiquetée.

Le client essaye de mettre des fanions STORE conditionnel sur un mélange de messages effacés et non effacés ; un message échoue à l'essai UNCHANGEDSINCE.

C: B001 STORE 1:7 (UNCHANGEDSINCE 320172338) +FLAGS (\SEEN)
S: * 1 FETCH (MODSEQ (320172342) FLAGS (\SEEN))
S: * 3 FETCH (MODSEQ (320172342) FLAGS (\SEEN))
S: B001 NO [MODIFIED 2] Certains des messages n'existent plus.
C: B002 NOOP
S: * 4 EXPUNGE
S: * 4 EXPUNGE
S: * 4 EXPUNGE
S: * 4 EXPUNGE
S: * 2 FETCH (MODSEQ (320172340) FLAGS (\Deleted \Answered))
S: B002 OK NOOP Completed.

En recevant les réponses FETCH pour les messages 1 et 3, et les réponse EXPUNGE qui indiquent que les messages 4 à 7 ont été effacés, le client ne réessaye l'opération que pour le message 2. La valeur UNCHANGEDSINCE mise à jour est utilisée.

C: b003 STORE 2 (UNCHANGEDSINCE 320172340) +FLAGS (\Seen)
S: * 2 FETCH (MODSEQ (320180050))
S: b003 OK Conditional Store completed

Note : Si un message est spécifié plusieurs fois dans l'ensemble de messages, et si le serveur n'élimine par les dupliqués en interne de l'ensemble de messages, il NE DOIT PAS échouer à l'opération STORE conditionnelle pour la seconde occurrence du message (ou les suivantes) si l'opération s'est achevée avec succès pour la première occurrence. Par exemple, si le client spécifie :

e105 STORE 7,3:9 (UNCHANGEDSINCE 12121230045) +FLAGS.SILENT (\Deleted)

le serveur ne doit pas faire échouer l'opération pour le message 7 au titre du traitement "3:9" si il a réussi quand le message 7 a été traité la première fois.

Une fois que le client a spécifié le modificateur UNCHANGEDSINCE dans une commande STORE, le serveur DOIT inclure les éléments de données de réponse à “aller chercher MODSEQ” dans toutes les réponses FETCH suivantes non sollicitées.

Le présent document change aussi le comportement du serveur quand il a effectué une commande STORE ou UID STORE et que le modificateur UNCHANGEDSINCE n'est pas spécifié. Si l'opération est réussie pour un message, le serveur DOIT mettre à jour l'attribut mod-sequence du message. Il est EXIGÉ du serveur d'inclure la valeur de mod-sequence chaque fois que il décide d'envoyer la réponse FETCH non sollicitée à tous les clients à capacité CONDSTORE qui ont ouvert la boîte aux lettres contenant le message.

Les mises en œuvre de serveur devraient aussi voir au paragraphe 3.8 les questions supplémentaires de qualité de mise en œuvre relatives à la commande STORE.

3.3 Commandes FETCH et UID FETCH

3.3.1 Modificateur CHANGEDSINCE FETCH

Le présent document définit le modificateur FETCH suivant (voir au paragraphe 2.4 de la [RFC4466]):

CHANGEDSINCE <mod-sequence>

Le modificateur CHANGEDSINCE de FETCH permet de créer un autre sous ensemble de la liste des messages décrits par l'ensemble de la séquence. Les informations décrites par les éléments de données de message ne sont retournées que pour les messages qui ont une mod-sequence plus grande que <mod-sequence>.

Quand le modificateur CHANGEDSINCE de FETCH est spécifié, il ajoute implicitement l'élément de données de message MODSEQ FETCH (paragraphe 3.3.2).

Exemple 12 :

C: s100 UID FETCH 1:* (FLAGS) (CHANGEDSINCE 12345)
S: * 1 FETCH (UID 4 MODSEQ (65402) FLAGS (\Seen))
S: * 2 FETCH (UID 6 MODSEQ (75403) FLAGS (\Deleted))
S: * 4 FETCH (UID 8 MODSEQ (29738) FLAGS ($NoJunk $AutoJunk$MDNSent))
S: s100 OK FETCH completed

3.3.2 Élément de données de message MODSEQ dans la commande FETCH

Cette extension ajoute un élément de données de message MODSEQ à la commande FETCH. L'élément de données de message MODSEQ permet aux clients de restituer les valeurs de mod-sequence pour une gamme de messages dans la boîte aux lettres actuellement choisie.

Une fois que le client a spécifié l'élément de données de message MODSEQ dans une demande FETCH, le serveur DOIT inclure les éléments de données de message MODSEQ dans toutes les réponses FETCH non sollicitées suivantes.

Syntaxe : MODSEQ

L'élément de données de message MODSEQ cause le retour par le serveur des éléments de données de réponse pour aller chercher MODSEQ.

Syntaxe : MODSEQ (<permsg-modsequence>)

Les éléments de données de réponse à MODSEQ contiennent les séquence de modification par message.

L'élément de données de réponse MODSEQ est retourné si le client a produit FETCH avec un élément de données de message MODSEQ. Il permet aussi au serveur de notifier au client les changements de mod-sequence causés par des STORE conditionnels (paragraphe 3.2) et/ou des changements causés par des sources externes.

Exemple 13 :

C: a FETCH 1:3 (MODSEQ)
S: * 1 FETCH (MODSEQ (624140003))
S: * 2 FETCH (MODSEQ (624140007))
S: * 3 FETCH (MODSEQ (624140005))
S: a OK Fetch complete

Dans cet exemple, le client demande des mod-sequences par message pour un ensemble de messages.

Quand un fanion pour un message est modifié dans une session différente, le serveur envoie une réponse FETCH non sollicitée contenant la mod-sequence pour le message.

Exemple 14 :

(Session 1, authentifiée comme l'utilisateur "alex"). L'utilisateur ajoute un fanion partagé \Deleted:

C: A142 SELECT INBOX
...
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Answered \Deleted \Seen *)] Limited
...
C: A160 STORE 7 +FLAGS.SILENT (\Deleted)
S: * 7 FETCH (MODSEQ (2121231000))
S: A160 OK Store completed

(Session 2, aussi authentifiée comme l'utilisateur "alex"). Tous les changements de fanions sont toujours rapportés à toutes les sessions authentifiées comme le même utilisateur comme dans la session 1.

C: C180 NOOP
S: * 7 FETCH (FLAGS (\Deleted \Answered) MODSEQ (12121231000))
S: C180 OK Noop completed

(Session 3, authentifiée comme utilisateur "andrew"). Comme \Deleted est un fanion partagé, les changements dans la session 1 sont aussi rapportés dans la session 3:

C: D210 NOOP
S: * 7 FETCH (FLAGS (\Deleted \Answered) MODSEQ (12121231000))
S: D210 OK Noop completed

L'utilisateur modifie un fanion privé \Seen dans la session 1...

C: A240 STORE 7 +FLAGS.SILENT (\Seen)
S: * 7 FETCH (MODSEQ (12121231777))

S: A240 OK Store completed

...qui est seulement rapporté dans la session 2...

C: C270 NOOP
S: * 7 FETCH (FLAGS (\Deleted \Answered \Seen) MODSEQ (12121231777))
S: C270 OK Noop completed

...mais pas dans la session 3.

C: D300 NOOP
S: D300 OK Noop completed

Et finalement, l'utilisateur retire les fanions \Answered (partagé) et \Seen (privé) dans la session 1.

C: A330 STORE 7 -FLAGS.SILENT (\Answered \Seen)
S: * 7 FETCH (MODSEQ (12121245160))
S: A330 OK Store completed

Les deux changements sont rapportés dans la session 2...

C: C360 NOOP
S: * 7 FETCH (FLAGS (\Deleted) MODSEQ (12121245160))
S: C360 OK Noop completed

...et seuls les changements aux fanions partagés sont rapportés dans la session 3.

C: D390 NOOP
S: * 7 FETCH (FLAGS (\Deleted) MODSEQ (12121245160))
S: D390 OK Noop completed

Les mises en œuvre de serveur devraient aussi voir au paragraphe 3.8 les questions supplémentaires de qualité de mise en œuvre relatives à la commande FETCH.

3.4 Critère de recherche MODSEQ dans SEARCH

Le critère MODSEQ pour la commande SEARCH permet au client de chercher les éléments de métadonnées qui ont été modifiés depuis un moment spécifié.

Syntaxe : MODSEQ [<nom d'entrée> <entry-type-req>] <mod-sequence-valzer>

Les messages qui ont des valeurs de modification égales ou supérieurs à <mod-sequence-valzer>. Cela permet au client, par exemple, de trouver quels messages contiennent des éléments de métadonnées qui ont changé depuis la dernière fois qu'il a mis à jour son antémémoire déconnectée. Le client peut aussi spécifier <nom d'entrée> (nom de l'élément de métadonnées) et <entry-type-req> (type de l'élément de métadonnées) avant <mod-sequence-valzer>. <entry-type-req> peut être "shared" *(partagé)*, "priv" *(privé)*, ou "all" *(tous)*. Ce dernier signifie que le serveur devrait utiliser la plus grande valeur parmi les séquence de modification "priv" et "shared" pour l'élément de métadonnées. Si le serveur ne mémorise pas en interne les séquences de modification séparées pour les différents éléments de métadonnées, il DOIT ignorer <nom d'entrée> et <entry-type-req>. Autrement, le serveur devrait les utiliser pour réduire la recherche.

Pour un fanion <flagname>, le <nom d'entrée> correspondant a une forme "/flags/<flagname>" comme défini dans la [RFC4466]. Noter que le caractère "\" en tête note un fanion système qui doit être échappé selon le paragraphe 4.3 de la [RFC3501], car <nom d'entrée> utilise cette syntaxe pour les chaînes entre guillemets.

Si le client spécifie un critère MODSEQ dans une commande SEARCH et si le serveur retourne un résultat SEARCH non vide, le serveur DOIT aussi ajouter (à la fin de la réponse SEARCH non étiquetée) la plus forte mod-sequence pour tous les messages retournés. Voir aussi le paragraphe 3.5.

Exemple 15 :

C: a SEARCH MODSEQ "/flags/\\draft" all 620162338
S: * SEARCH 2 5 6 7 11 12 18 19 20 23 (MODSEQ 917162500)
S: a OK Search complete

Dans l'exemple ci-dessus, le numéro de message de tout message contenant la chaîne "IMAP4" dans l'attribut "value" de l'entrée "/comment" qui a une mod-sequence égale ou supérieure à 620162338 pour le fanion "\Draft" est retourné dans les résultats de recherche.

Exemple 16 :

C: t SEARCH OR NOT MODSEQ 720162338 LARGER 50000
S: * SEARCH
S: t OK Recherche achevée, aucun résultat trouve

3.5 Réponse modifiée SEARCH non étiquetée

Données : zéro, un ou plusieurs numéros de valeur de mod-sequence (omis si aucune correspondance)

Le présent document étend la syntaxe de la réponse SEARCH non étiquetée pour inclure la plus forte mod-sequence pour tous les messages retournés.

Si un client spécifie un critère MODSEQ dans une commande SEARCH (ou UID SEARCH) et si le serveur retourne un résultat SEARCH non vide, le serveur DOIT aussi ajouter (à la fin de la réponse SEARCH non étiquetée) la plus forte mod-sequence pour tous les messages retournés. Voir les exemples au paragraphe 3.4.

3.6 Éléments de données d'état HIGHESTMODSEQ

Le présent document définit un nouvel élément de données d'état : HIGHESTMODSEQ

La plus forte valeur de mod-sequence de tous les messages dans la boîte aux lettres. C'est la même valeur que celle retournée par le serveur dans le code de réponse HIGHESTMODSEQ dans une réponse OK non étiquetée (voir au paragraphe 3.1.1). Si le serveur ne prend pas en charge la mémorisation persistante des mod-sequences pour la boîte aux lettres (voir au paragraphe 3.1.2) le serveur DOIT retourner 0 comme valeur des élément de données d'état HIGHESTMODSEQ.

Exemple 17 :

C: A042 STATUS blurdybloop (UIDNEXT MESSAGES HIGHESTMODSEQ)
S: * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292 HIGHESTMODSEQ 7011231777)
S: A042 OK STATUS completed

3.7 Paramètre CONDSTORE pour SELECT et EXAMINE

L'extension CONDSTORE définit un seul paramètre choisi facultatif, "CONDSTORE", qui dit au serveur qu'il DOIT inclure les éléments de données de réponse d'aller chercher MODSEQ dans toutes les réponses FETCH non sollicitées suivantes.

Le paramètre CONDSTORE à SELECT/EXAMINE aide à éviter une condition de compétition qui pourrait se produire quand un ou plusieurs éléments de métadonnées sont modifiés dans une autre session après que le serveur a envoyé le code de réponse HIGHESTMODSEQ et avant que le client soit capable de produire une commande activant CONDSTORE.

Exemple 18 :

C: A142 SELECT INBOX (CONDSTORE)
S: * 172 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Le message 12 est le premier non vu
S: * OK [UIDVALIDITY 3857529045] UID valides
S: * OK [UIDNEXT 4392] Prochain UID prévu

S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen *)] Limited
S: * OK [HIGHESTMODSEQ 715194045007]
S: A142 OK [READ-WRITE] SELECT achevé, CONDSTORE est maintenant activé

3.8 Questions supplémentaires sur la qualité de mise en œuvre

Les mises en œuvre de serveur devraient suivre la règle qui suit, qui s'applique à toute commande STORE/UID STORE (avec et sans modificateur UNCHANGEDSINCE) achevée avec succès, ainsi qu'à une commande FETCH qui établit implicitement le fanion \Seen.

Ajouter le fanion quand il est déjà présent ou le supprimer quand il n'est pas présent NE DEVRAIT PAS changer la mod-sequence. Cela va empêcher les demandes parasites de synchronisation du client.

Cependant, noter que les mises en œuvre de client NE DOIVENT PAS compter sur ce comportement du serveur. Un client ne peut pas distinguer entre le cas où un serveur a violé le DEVRAIT mentionné ci-dessus, et celui où un ou plusieurs clients établissent et suppriment (ou suppriment et établissent) le fanion dans une autre session.

4. Syntaxe formelle

La spécification de syntaxe suivante utilise la notation en forme Backus-Naur augmentée (ABNF) [RFC4234]. Les éléments non définis ici peuvent être trouvés dans la syntaxe formelle des spécifications de l'ABNF [RFC4234], de IMAP [RFC3501], et des extensions IMAP ABNF [RFC4466].

Sauf notation contraire, tous les caractères alphabétiques sont insensibles à la casse. L'utilisation de caractères majuscules ou minuscules pour définir des chaînes de jetons est seulement pour la clarté rédactionnelle. Les mises en œuvre DOIVENT accepter ces chaînes de façon insensible à la casse.

capability =/ "CONDSTORE"

status-att =/ "HIGHESTMODSEQ"
;; étend le non-terminal défini dans la RFC 3501.

status-att-val =/ "HIGHESTMODSEQ" SP mod-sequence-valzer
;; étend le non-terminal défini dans la [RFC4466]. La valeur 0 note que la boîte aux lettres ne prend pas en charge la mémorisation persistante des séquences de modification comme décrit au paragraphe 3.1.2

store-modifier =/ "UNCHANGEDSINCE" SP mod-sequence-valzer
;; Un seul "UNCHANGEDSINCE" peut être spécifié dans une opération STORE

fetch-modifier =/ chgsince-fetch-mod
;; conforme à la syntaxe générique de "fetch-modifier" définie dans la [RFC4466].

chgsince-fetch-mod = "CHANGEDSINCE" SP mod-sequence-value
;; Le modificateur CHANGEDSINCE FETCH se conforme à la syntaxe de fetch-modifier

fetch-att =/ fetch-mod-sequence
;; modifie le fetch-att IMAP4 original

fetch-mod-sequence = "MODSEQ"

fetch-mod-resp = "MODSEQ" SP "(" permsg-modsequence ")"

msg-att-dynamic =/ fetch-mod-resp

search-key =/ search-modsequence
;; modifie le search-key IMAP4 original. Ce changement s'applique à toutes les commandes qui se réfèrent à ce non terminal, en particulier SEARCH.

search-modsequence = "MODSEQ" [search-modseq-ext] SP mod-sequence-valzer

search-modseq-ext = SP nom d'entrée SP entry-type-req

resp-text-code =/ "HIGHESTMODSEQ" SP mod-sequence-value / "NOMODSEQ" / "MODIFIED" SP set

nom d'entrée = entry-flag-name

entry-flag-name = DQUOTE "/flags/" attr-flag DQUOTE
;; chaque fanion défini par le système ou l'utilisateur <flag> est transposé en "/flags/<flag>". <entry-flag-name> suivant les règles d'échappement utilisées par une chaîne "guillemettée" comme décrit au paragraphe 4.3 de la [RFC3501], par exemple, pour le fanion \Seen le <nom d'entrée> correspondant est "/flags/\\seen", et pour le fanion $MDNSent, le <nom d'entrée> correspondant est "/flags/$mdnsent".

entry-type-resp = "priv" / "shared"
;; type d'élément de métadonnées

entry-type-req = entry-type-resp / "all"
;; effectue l'opération SEARCH sur un élément de métadonnées privé, un élément de métadonnées partagé ou les deux.

permsg-modsequence = mod-sequence-value
;; mod-sequence par message

mod-sequence-value = 1*DIGIT
;; Entier positif non signé de 64 bit s (mod-sequence) (1 ≤ n < 18 446 744 073 709 551 615)

mod-sequence-valzer = "0" / mod-sequence-value

search-sort-mod-seq = "(" "MODSEQ" SP mod-sequence-value ")"

select-param =/ condstore-param
;; Se conforme à la syntaxe générique de non terminal "select-param" définie dans la [RFC4466].

condstore-param = "CONDSTORE"

maiolbox-data =/ "SEARCH" [1*(SP nz-number) SP search-sort-mod-seq]

attr-flag = "\\Answered" / "\\Flagged" / "\\Deleted" / "\\Seen" / "\\Draft" / attr-flag-keyword / attr-flag-extension
;; n'inclut pas "\\Recent"

attr-flag-extension = "\\" atom
;; Future expansion. Les mises en œuvre de client DOIVENT accepter les fanions flag-extension. Les mises en œuvre de serveur NE DOIVENT PAS générer de fanions flag-extension sauf comme défini par de futures révisions normalisées ou sur la voie de la normalisation de la [RFC3501].

attr-flag-keyword = atom

5. Considérations sur les mises en œuvre de serveur

Cette Section décrit comment une mise en œuvre de serveur qui ne mémorise pas les séquences de modification séparées par métadonnées pour différents éléments de métadonnées peut éviter d'envoyer la réponse MODIFIED à une des opérations STORE conditionnelles suivantes :
+FLAGS
-FLAGS
+FLAGS.SILENT
-FLAGS.SILENT

Noter que l'optimisation décrite dans cette section ne peut pas être effectuée dans le cas d'une opération STORE FLAGS conditionnelle.

Par exemple, le client a produit :

C: a106 STORE 100:150 (UNCHANGEDSINCE 212030000000) +FLAGS.SILENT ($Processed)

Quand le serveur reçoit la commande et l'analyse avec succès, il itère à travers l'ensemble de messages et essaye d'exécuter la commande STORE conditionnelle pour chaque message.

Chaque serveur travaille en interne comme un client, c'est-à-dire, il doit mettre en antémémoire l'état actuel de tous les fanions IMAP qui sont connus du client. Afin de rapporter les changements de fanions au client, le serveur compare les valeurs en antémémoire aux valeurs de sa base de données pour les fanions IMAP.

Imaginons qu'un autre client ait changé l'état d'un fanion \Deleted sur le message 101 et que le changement ait mis à jour la mod-sequence pour le message. Le serveur sait que la mod-sequence pour la boîte aux lettres a changé ; cependant, il sait aussi que :
a) le client n'est pas intéressé par le fanion \Deleted , car il ne l'a pas inclus dans son opération +FLAGS.SILENT; et
b) l'état du fanion $Processed n'a pas changé (le serveur peut le déterminer en comparant l'état du fanion en antémémoire avec l'état du fanion dans la base de données).

Donc, le serveur n'a pas à rapporter MODIFIED au client. À la place, le serveur peut établir le fanion $Processed, mettre à jour la mod-sequence pour le message 101 une fois encore et envoyer une réponse FETCH non étiquetée avec une nouvelle mod-sequence et les fanions :

S: * 101 FETCH (MODSEQ (303011130956) FLAGS ($Processed \Deleted \Answered))

Voir aussi au paragraphe 3.8 sur les problémes supplémentaires de qualité de mise en œuvre.

6. Considérations sur la sécurité

Il est estimé que l'extension conditionnelle STORE ne soulève aucun nouveau souci de sécurité qui ne soit déjà discuté dans la [RFC3501]. Cependant, la disponibilité de cette extension peut rendre possible une utilisation de IMAP4 dans des applications critiques qui n'aurait pas été possible précédemment, rendant encore plus importante une mise en œuvre et un fonctionnement corrects du serveur IMAP.

7. Considérations relatives à l'IANA

Les capacités IMAP4 sont enregistrées en publiant une RFC sur la voie de la normalisation ou une RFC expérimentale approuvée parl'IESG. Le registre est actuellement situé à : http://www.iana.org/assignments/imap4-capabilities

Le présent document définit la capacité CONDSTORE pour IMAP. L'IANA l'a ajoutée en conséquence au registre.

8. Références

8.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC3501] M. Crispin, "Protocole d'accès au message Internet - version 4rev1", mars 2003. *(P.S. ; MàJ par* RFC4466, 4469, 4551, 5032, 5182, 7817, 8314, 8437, 8474)

[RFC4234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", octobre 2005. *(Remplace* RFC2234, *remplacée par* RFC5234)

[RFC4466] A. Melnikov, C. Daboo, "Récapitulation des extensions à l'ABNF pour IMAP4", avril 2006. *(P.S.)*

8.2 Références pour information

[RFC1305] D. Mills, "Protocole de l'heure du réseau, version 3, spécification, mise en œuvre et analyse", STD 12, mars992. *(Remplacée par* RFC5905)

[RFC2180] M. Gahrns, "Pratique de boîtes aux lettres à accès multiple dans IMAP4", juillet 1997. *(Information)*

[RFC2244] C. Newman, J. G. Myers,"ACAP – Protocole d'accès à la configuration d'application", novembre 1997. *(P.S.)*

[RFC4314] A. Melnikov, "Extension IMAP4 de liste de contrôle d'accès (ACL)", décembre 2005. (*Remplace* RFC2086) *(P.S.)*

[RFC5257] C. Daboo, R. Gellens, "Protocole d'accès au message Internet - extension ANNOTATE", juin 2008. *(Exp.)*

9. Remerciements

Du texte a été emprunté à "Extension IMAP ANNOTATE" [RFC5257] de Randall Gellens et Cyrus Daboo et à "ACAP – Protocole d'accès à al configuration d'application" [RFC2244] de Chris Newman et John Myers.

Tous nos remerciements à Randall Gellens pour sa relecture attentive de ce document.

Les auteurs remercient aussi de leurs retours Cyrus Daboo, Larry Greenfield, Chris Newman, Harrie Hazewinkel, Arnt Gulbrandsen, Timo Sirainen, Mark Crispin, Ned Freed, Ken Murchison, et Dave Cridland.

Adresse des auteurs

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex
TW12 2BX,
United Kingdom
mél : Alexey.Melnikov@isode.com

Steve Hole
ACI WorldWide/MessagingDirect
#1807, 10088 102 Ave
Edmonton, AB
T5J 2Z1
Canada
mél : Steve.Hole@messagingdirect.com

Déclaration complète de droits de reproduction

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.