Groupe de travail Réseau
Request for Comments : 2244
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

C. Newman, Innosoft
J. G. Myers, Netscape
novembre 1997

ACAP – Protocole d'accès à la configuration d'application

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le protocole d'accès à la configuration d'application (ACAP, *Application Configuration Access Protocol*) est conçu pour prendre en charge la mémorisation et l'accès à distance des informations d'option de programme, de configuration et de préférences. Le modèle de mémorisation des données est conçu pour permettre à un client un accès relativement simple aux données intéressantes, à permettre que de nouvelles informations soient facilement ajoutées sans reconfiguration du serveur, et à promouvoir l'utilisation aussi bien de données normalisées que de données propriétaires. Les caractéristiques clés incluent "l'héritage" qui peut être utilisé pour gérer les valeurs par défaut pour l'établissement de la configuration et de listes de contrôle d'accès qui permettent de partager des informations personnelles intéressantes et d'interdire l'accès à des informations de groupe.

Table des matières

1. Introduction

1.1 Conventions utilisées dans ce document

Dans les exemples, "C:" et "S:" indiquent respectivement les lignes envoyées par le client et par le serveur. Si de telles lignes reviennent à la ligne sans une nouvelle étiquette "C:" ou "S:", le retour à la ligne est motivé par un souci de présentation et ne fait pas partie de la commande.

Les mots clés "EXIGE", "DOIT", "NE DOIT PAS", "DEVRAIT", "NE DEVRAIT PAS", et "PEUT" dans ce document sont à interpréter comme décrit dans "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence" [RFC2119].

1.2 Modèle de données ACAP

Un serveur ACAP exporte une arborescence d'entrées hiérarchisées. Chaque niveau de l'arborescence est appelée un dataset *(ensemble de données)* et chaque ensemble de données est constitué d'une liste d'entrées. Chaque entrée a un nom univoque et peut contenir un nombre arbitraire d'attributs désignés. Chaque attribut au sein d'une entrée peut être d'une seule valeur ou de plusieurs valeurs et peut avoir des métadonnées associées pour aider à l'accès et à l'interprétation de la valeur.

Les règles avec lesquelles un client interprète les données au sein d'une portion d'une arborescence d'entrées ACAP sont appelées une classe d'ensemble de données *(dataset class)*.

1.3 Objectifs de conception d'ACAP

Le principal objet d'ACAP est de permettre aux usagers d'accéder à leurs données de configuration à partir de plusieurs ordinateurs connectés au réseau. Les usagers peuvent alors se mettre devant n'importe quel ordinateur connecté au réseau, en utilisant n'importe quelle application à capacité ACAP, et avoir accès à leurs propres données de configuration. Comme on espère que de nombreuses applications auront la capacité ACAP, la simplicité du client a été préférée à celle du serveur ou du protocole chaque fois que c'était raisonnable.

ACAP est conçu pour être facilement gérable. Pour cette raison, il inclut "l'héritage" qui permet à un ensemble de données d'hériter des attributs par défaut provenant d'un autre ensemble de données. De plus, des listes de contrôle d'accès sont incluses pour permettre à des délégations de gestion et à des quotas d'être inclus dans les mémorisations de contrôle. Enfin, un serveur ACAP qui se conforme à la présente spécification de base devrait être capable de prendre en charge la plupart des classes d'ensembles de données qui seront définies à l'avenir sans exiger une reconfiguration ou mise à niveau du serveur.

ACAP est conçu pour bien fonctionner avec un client qui n'a qu'un accès intermittent à un serveur ACAP. Pour cette raison, chaque entrée a une heure de modification conservée par le serveur afin que le client puisse détecter les changements. De plus, le client peut demander au serveur la liste des entrées qui ont été retirées depuis son dernier accès au serveur.

ACAP présuppose qu'un ensemble de données peut éventuellement être grand et/ou que la connexion du client au réseau peut être lente, et offre donc un tri des serveurs, une collecte sélective et une notification des changements des entrées au sein d'un ensemble de données.

Comme nécessaire pour la plupart des protocoles de l'Internet, la sécurité, l'adaptabilité et l'internationalisation étaient des objectifs importants de la conception.

Ces objectifs de conception étant donnés, il a été tenté de garder ACAP aussi simple que possible. C'est un protocole Internet traditionnel fondé sur le texte qui rend très simple le débogage de protocole. Il a été conçu à partir du cadre du protocole IMAP [RFC2060] qui a connu un grand succès, avec quelque raffinements.

1.4 Validation

Par défaut, toute valeur peut être mémorisée dans tout attribut pour lequel l'usager a les permissions et quotas appropriés. Cette règle est nécessaire pour permettre l'ajout de nouvelles classes simples d'ensembles de données sans reconfigurer ou mettre à niveau le serveur.

Dans certains cas, comme lorsque la valeur a une signification particulière pour le serveur, il est utile que le serveur effectue une validation en retournant le code de réponse INVALID à une commande STORE. Ces cas DOIVENT être explicitement identifiés dans la spécification de classe d'ensemble de données qui DEVRAIT inclure des règles fixées spécifiquement pour la validation. Comme un serveur ACAP particulier peut n'être pas au courant d'une spécification de classe d'ensemble de données particulière, les clients NE DOIVENT PAS s'appuyer sur la présence d'une application de validation chez le serveur.

1.5 Définitions

Liste de contrôle d'accès (ACL, *access control list*)
C'est un ensemble de paires d'identifiant et de droits associé à un objet. Une ACL sert à déterminer quelles opérations sont permises à un usager sur cet objet. Voir le paragraphe 3.5.

attribut
C'est une valeur désignée au sein d'une entrée. Voir le paragraphe 3.1.

comparateur
C'est une fonction désignée qui peut être utilisée pour effectuer une ou plusieurs des trois opérations de comparaison : rangement, comparaison d'égalité et correspondance de sous-chaîne. Voir le paragraphe 3.4.

contexte
C'est un ensemble ordonné d'entrées dans un ensemble de données, créé par une commande SEARCH avec un modificateur MAKECONTEXT. Voir le paragraphe 3.3.

ensemble de données
C'est un niveau de la hiérarchie dans l'arborescence des entrées de ACAP.

spécification de classe d'ensemble de données
Ce sont les règles qui permettent à un client d'interpréter les données dans une portion de l'arborescence des entrées de ACAP.

entrée
C'est un ensemble d'attributs avec un nom d'entrée unique. Voir le paragraphe 3.1.

metadonnées
Ce sont des informations qui décrivent un attribut, sa valeur et tous les contrôles d'accès associés à cet attribut. Voir le paragraphe 3.1.2.

NIL Cela représente la non existence d'un élément de données particulier.

NUL C'est un caractère de contrôle codé par 0 en [US-ASCII].

octet C'est une valeur de 8 bits. Sur la plupart des systèmes d'ordinateur modernes, un octet est de 8 bits.

SASL Authentification simple et couche de sécurité *(Simple Authentication and Security Layer)* [RFC2222].

UTC *(Universal Coordinated Time)* Temps universel tel que conservé par le Bureau International des Poids et Mesures (BIPM).

UTF-8
C'est un format de transformation en 8 bits de l'ensemble universel de caractères [RFC2044]. Noter qu'un changement incompatible a été fait à l'ensemble de caractères codés référencé par la [RFC2044], de sorte que pour les besoins du présent document, UTF-8 se réfère au codage UTF-8 tel que défini dans la version 2.0 de Unicode [UNICODE-2], ou de [ISO-10646] y compris les amendements un à sept.

1.6 Généralités sur les commandes ACAP

Les commandes AUTHENTICATE, NOOP, LANG et LOGOUT fournissent les services de base du protocole. La commande SEARCH est utilisée pour choisir, trier, aller chercher et surveiller les changements aux valeurs des attributs et aux métadonnées. Les commandes UPDATECONTEXT et FREECONTEXT sont aussi utilisées pour aider à la surveillance des changements des valeurs des attributs et des métadonnées. La commande STORE est utilisée pour ajouter, modifier et supprimer des entrées et des attributs. La commande DELETEDSINCE est utilisée pour aider un client à re-synchroniser une antémémoire avec le serveur. Les commandes GETQUOTA, SETACL, DELETEACL, LISTRIGHTS et MYRIGHTS sont utilisées pour examiner les quotas de mémorisation et examiner ou modifier les permissions d'accès.

2. Cadre du protocole

2.1 Niveau liaison

Le protocole ACAP suppose un flux de données fiable tel que fourni par TCP. Lorsque TCP est utilisé, un serveur ACAP écoute sur l'accès 674.

2.2 Commandes et réponses

Une session ACAP consiste en l'établissement d'une connexion client/serveur, un accueil initial de la part du serveur, et des interactions client/serveur. Ces interactions client/serveur consistent en une commande du client, des données du serveur et un résultat final du serveur.

ACAP est un protocole fondé sur le texte en mode ligne. En général, les interactions transmises par les clients et les serveurs sont sous forme de lignes ; c'est-à-dire, des séquences de caractères qui se terminent par un CRLF. Le receveur de protocole d'un client ou serveur ACAP est soit en train de lire une ligne, soit en train de lire une séquence d'octets avec un compte connu (un littéral) suivi par une ligne. Les clients et les serveurs doivent tous deux être capables de traiter des lignes de toutes longueurs.

2.2.1 Protocole client envoyeur et protocole serveur receveur

La commande client commence une opération. Chaque commande client est préfixée d'un identifiant (une chaîne alphanumérique de pas plus de 32 caractères, par exemple, A0001, A0002, etc.) appelée une "étiquette". Une étiquette différente DEVRAIT être générée par le client pour chaque commande.

Il y a deux cas dans lesquels une ligne du client ne représente pas une commande complète. Dans un des cas, un argument de commande est cité avec un compte d'octets (voir la description du littéral au paragraphe 2.6.3) ; dans l'autre cas, l'argument de commande exige un retour du serveur (voir la commande AUTHENTICATE). Dans certains de ces cas, le serveur envoie une demande de continuation de commande si il est prêt pour la partie suivante de la commande. Cette réponse est munie en préfixe du jeton "+".

Note : Si, à la place, le serveur a détecté une erreur dans une 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'envoyer la suite de la commande.

Il est aussi possible au serveur d'envoyer une réponse d'achèvement ou une réponse intermédiaire pour quelque autre

commande (si plusieurs commandes sont en cours) ou des données non étiquetées. Dans l'un ou l'autre cas, la demande de continuation de commande est toujours en instance ; le client prend les mesures appropriées pour la réponse, et lit une autre réponse venant du serveur.

Le serveur ACAP lit une ligne de commande provenant du client, analyse la commande et ses arguments, et transmet les données du serveur et un résultat d'achèvement de commande du serveur.

2.2.2 Protocole serveur envoyeur et protocole client receveur

Les données transmises du serveur au client viennent sous quatre formes :
- demandes de continuation de commande,
- résultats d'achèvement de commande,
- réponses intermédiaires,
- réponses non étiquetées.

Une demande de continuation de commande est préfixée du jeton "+".

Un résultat d'achèvement de commande indique le succès ou l'échec de l'opération. Il est étiqueté avec la même étiquette que la commande de client qui commence l'opération. Donc, si plus d'une commande est en cours, l'étiquette dans une réponse d'achèvement d'un serveur identifie la commande à laquelle la réponse s'applique. Il y a trois réponses d'achèvement de serveur possibles : OK (qui indique le succès) : NON (qui indique l'échec) ; ou BAD (qui indique une erreur de protocole comme une commande non reconnue ou une erreur de syntaxe de commande).

Une réponse intermédiaire retourne des données qui ne peuvent être interprétées que dans le contexte d'une commande en cours. Elle est étiquetée avec la même étiquette que la commande client qui a commencé l'opération. Donc, si plus d'une commande est en cours, l'étiquette d'une réponse intermédiaire identifie la commande à laquelle s'applique la réponse. Une réponse étiquetée autre que "OK", "NON", ou "BAD" est une réponse intermédiaire.

Une réponse non étiquetée retourne des données ou des messages d'état qui peuvent être interprétés en dehors du contexte d'une commande en cours. Elles est préfixée avec le jeton "*". Les données non étiquetées peuvent être envoyées par suite d'une commande client, ou peuvent être envoyées de façon unilatérale par le serveur. Il n'y a pas de différence syntaxique entre les données non étiquetées qui résultent d'une commande spécifique et les données non étiquetées qui ont été envoyées de façon unilatérale.

Le receveur de protocole d'un client ACAP lit une ligne de réponse provenant du serveur. Il décide alors de l'action sur la réponse sur la base du premier jeton de la réponse, qui peut être une étiquette, une "*", ou un "+" comme décrit ci-dessus.

Un client DOIT être prêt à accepter toute réponse de serveur à tout moment. Cela inclut les données non étiquetées qu'il peut n'avoir pas demandées.

Cette question est exposée plus en détails au paragraphe sur les réponses du serveur.

2.3 États du serveur

Un serveur ACAP peut prendre l'un des trois états décrits ci-après. La plupart des commandes ne sont valides que dans certains états. C'est une erreur de protocole pour le client de tenter une commande alors que le serveur est dans un état inapproprié pour cette commande. Dans ce cas, un serveur va répondre avec un résultat d'achèvement de commande BAD.

2.3.1 État non authentifié

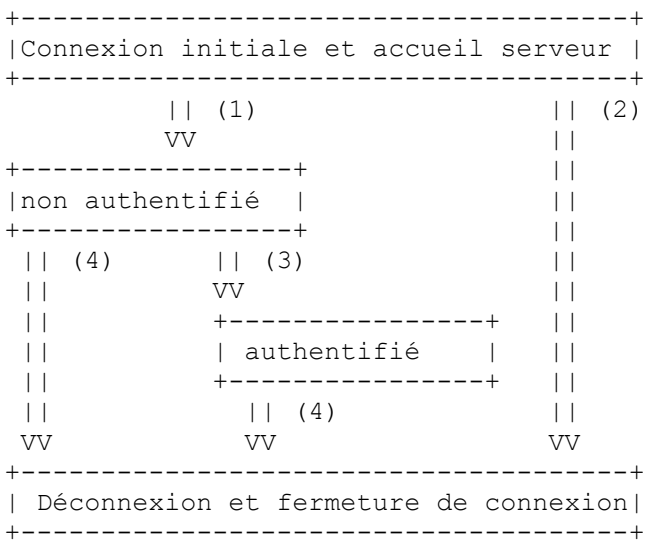
Dans l'état non authentifié, l'usager doit fournir des accréditifs d'authentification avant que la plupart des commandes soient permises. On entre dans cet état au début d'une connexion.

2.3.2 État authentifié

Dans l'état authentifié, l'usager est authentifié et la plupart des commandes vont être permises. On entre dans cet état lorsque des accréditifs d'authentification acceptables ont été fournis.

2.3.3 État déconnecté

Dans l'état déconnecté, la session est en cours de clôture, et le serveur va clore la connexion. Cet état peut survenir par suite d'une demande du client ou par une décision unilatérale du serveur.



(1) connexion (accueil ACAP)
(2) connexion rejetée (accueil BYE)
(3) commande AUTHENTICATE réussie
(4) commande LOGOUT, clôture du serveur ou clôture de connexion

2.4 Considérations sur le fonctionnement

2.4.1 Mise à jour d'état non étiquetées

À tout moment, un serveur peut envoyer des données que le client n'a pas demandées.

2.4.2 Réponse quand aucune commande n'est en cours

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

2.4.3 Temporisateur d'auto déconnexion

Si un serveur a un temporisateur d'auto déconnexion d'inactivité, ce temporisateur DOIT être d'une durée d'au moins 30 minutes. La réception d'une commande ANY de la part du client durant cet intervalle DOIT suffire à remettre à zéro de temporisateur d'auto déconnexion.

2.4.4 Commandes multiples en cours

Le client n'est pas obligé d'attendre le résultat de l'achèvement d'une commande avant d'envoyer une autre commande, sous réserve des contraintes de commandes de flux du flux de données sous-jacent. De même, un serveur n'est pas obligé de traiter entièrement une commande avant de commencer le traitement de la commande suivante, sauf si une ambiguïté devait résulter d'une commande qui affecterait le résultat des autres commandes. Si il y a une telle ambiguïté, le serveur exécute les commandes jusqu'à achèvement dans l'ordre donné par le client.

2.5 Demande de continuation de commande du serveur

La demande de continuation de commande est indiquée par un jeton "+" au lieu d'une étiquette. Cela indique que le serveur est prêt à accepter la continuation d'une commande provenant du client.

Cette réponse est utilisée dans la commande AUTHENTICATE pour transmettre des données du serveur au client, et pour demander des données supplémentaires du client. Cette réponse est aussi utilisée si un argument de toute commande est un littéral de synchronisation (voir au paragraphe 2.6.3).

Il n'est pas permis au client d'envoyer les octets d'un littéral de synchronisation sauf si le serveur indique qu'il l'attend. Cela permet au serveur de traiter les commandes et de rejeter les erreurs ligne par ligne, en supposant qu'il vérifie qu'il n'y a pas de littéraux non synchronisants à la fin de chaque ligne. Le reste de la commande, y compris le CRLF qui termine une commande, suit les octets du littéral. Si il y a des arguments de commande supplémentaires, les octets littéraux sont suivis par une espace et ces arguments.

Exemple :
C: A099 FREECONTEXT {10}
S: + "Prêt pour du texte de commande supplémentaire"
C: FRED
C: FOOB
S: A099 OK "FREECONTEXT terminé"
C: A044 BLURDYBLOOP {102856}
S: A044 BAD "Pas de commande telle que 'BLURDYBLOOP'"

2.6 Formats des données

ACAP utilise des commandes et réponses textuelles. Les données dans ACAP peuvent être d'une des cinq formes suivantes : atome, nombre, chaîne, liste entre parenthèses, ou NIL.

2.6.1 Atome

Un atome comporte de un à 1024 caractères non spéciaux . Il doit commencer par une lettre. Les atomes sont utilisés pour les mots clés du protocole.

2.6.2 Nombre

Un nombre comporte un ou plusieurs caractères de chiffres, et représente une valeur numérique. Les nombres sont restreints à la gamme des entiers non signés de 32 bits : 0 < nombre < 4 294 967 296.

2.6.3 Chaîne

Une chaîne a une des deux formes suivantes : littérale et entre guillemets (ou guillemetée). La forme littérale est la forme générale d'une chaîne. La forme entre guillemets est une solution de remplacement qui évite la redondance de traitement d'une chaîne littérale au prix de restrictions sur ce qui peut entrer dans une chaîne entre guillemets.

Une chaîne littérale est une séquence de zéro, un, ou plusieurs octets (incluant CR et LF), un préfixe avec un compte d'octets sous la forme d'une accolade ouverte ("{"), le nombre d'octets, l'accolade fermée ("}"), et le CRLF. Dans le cas de littéraux transmis du serveur au client, le CRLF est immédiatement suivi par les octets de données.

Il y a deux formes de littéraux transmis du client au serveur. La forme où l'accolade ouverte ("{") et le nombre d'octets sont immédiatement suivis par une accolade fermée ("}") et un CRLF est appelée un littéral synchronisant. Lors de l'envoi d'un littéral synchronisant, le client doit attendre de recevoir une demande de continuation de commande avant d'envoyer les octets de données (et le reste de la commande). L'autre forme de littéral, le littéral non synchronisant, est utilisé pour transmettre une chaîne du client au serveur sans attendre une demande de continuation de commande. Le littéral non synchronisant diffère du littéral synchronisant par un plus ("+") entre le nombre d'octets et l'accolade fermée ("}") et par les octets de données qui suivent immédiatement le CRLF.

Une chaîne guillemetée est une séquence de zéro à 1024 octets excluant NUL, CR et LF, avec des guillemets (<">) à chaque bout.

La chaîne vide est représentée par "" (une chaîne guillemetée avec zéro caractère entre les guillemets), comme {0} suivi par un CRLF (un littéral synchronisant avec un compte d'octet de 0), ou par {0+} suivi par un CRLF (un littéral non synchronisant avec un compte d'octet de 0).

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

2.6.3.1 Chaîne à 8 bits et binaires

La plupart des chaînes dans ACAP sont restreintes aux caractères UTF-8 et ne doivent pas contenir d'octets NUL. Les valeurs d'attribut PEUVENT contenir tous octets y compris NUL.

2.6.4 Listes entre parenthèses

Les structures de données sont représentées comme des "listes entre parenthèses" ; une séquence d'éléments de données, délimitée par une espace, et bordée à chaque bout par des parenthèses. Une liste entre parenthèses peut contenir d'autres listes entre parenthèses, en utilisant plusieurs niveaux de parenthèses pour indiquer l'incorporation.

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

2.6.5 NIL

L'atome spécial "NIL" représente la non existence d'un élément de données particulier qui est représenté par une chaîne ou une liste entre parenthèses, et est distinct de la chaîne vide "" ou de la liste entre parenthèses vide ().

3. Éléments du protocole

La présente section définit les formats des données et des autres éléments de protocole utilisés tout au long du protocole ACAP.

3.1 Entrées et attributs

Au sein d'un ensemble de données, chaque nom d'entrée est constitué de zéro, un ou plusieurs caractères UTF-8 autres qu'une barre oblique ("/"). Une liste d'entrées séparées par des barres obliques, une à chaque niveau de la hiérarchie, forme le chemin complet d'une entrée.

Chaque entrée est constituée d'un ensemble d'attributs. Chaque attribut a un nom hiérarchique en UTF-8, avec chaque composant du nom séparé par un point (".").

La valeur d'un attribut est d'une seule ou de plusieurs valeurs. Une seule valeur est NIL (n'a pas de valeur) ou une chaîne de zéro, un ou plusieurs octets. Une valeur multiple est une liste de zéro, une ou plusieurs chaînes, chacune comportant zéro, un ou plusieurs octets.

Il n'est pas permis aux noms d'attribut de contenir d'astérisque ("*") ou de pourcentage ("%") et ils DOIVENT être des chaînes UTF-8 valides qui ne contiennent pas NUL. Des noms d'attribut invalides résultent en une réponse BAD. Il n'est pas permis aux noms d'entrée de commencer par un point "." ou de contenir des barres obliques ("/") et ils DOIVENT être des chaînes UTF-8 valides qui ne contiennent pas NUL. Les noms d'entrée invalides dans le champ d'entrée d'une commande résultent en une réponse BAD.

L'utilisation de caractères UTF-8 non visibles dans les noms d'attribut et d'entrées est déconseillée.

3.1.1 Attributs prédéfinis

Les noms d'attribut qui ne contiennent pas un point (".") sont réservés aux attributs normalisés qui ont une signification dans tous les ensembles de données. Les attributs suivants sont définis par le protocole ACAP.

entrée
Contient le nom de l'entrée. Elle DOIT être d'une seule valeur. Les tentatives d'utiliser des valeurs illégales ou à valeurs multiples pour les attributs d'entrée sont des erreurs de protocole et DOIVENT résulter en une réponse d'achèvement BAD. Ceci est un cas particulier.

modtime *(heure de modification)*

Contient la date et l'heure de la dernière modification de toutes métadonnées en lecture-écriture de l'entrée. Cette valeur DOIT être en UTC, et DOIT être automatiquement mise à jour par le serveur.
La valeur consiste en 14 chiffres US-ASCII ou plus. Les quatre premiers indiquent l'année, les deux suivants indiquent le mois, les deux suivants indiquent le jour du mois, les deux suivants indiquent l'heure (de 0 à 23) les deux suivants indiquent la minute, et les deux suivants indiquent la seconde. Tout chiffre ultérieur indique les fractions de seconde.
L'heure, en particulier les fractions de seconde, n'a pas besoin d'être exactes. Il est EXIGÉ cependant que tout couple d'entrées d'un ensemble de données changé par des modifications successives ait des valeurs de modtime strictement ascendantes. De plus, chaque commande STORE au sein d'un ensemble de données (y compris des mémorisations simultanées provenant de différentes connexions) DOIT utiliser des valeurs de modtime différentes.
Cet attribut a une validation forcée, de sorte que toute tentative de mémoriser avec STORE une valeur dans cet attribut PEUT résulter en une réponse NO avec un code de réponse INVALID.

subdataset *(sous-ensemble de données)*
Si cet attribut est établi, cela indique l'existence d'un sous-ensemble de données dans cette entrée.

La valeur consiste en une liste d'URL relatifs ACAP (voir au paragraphe 3.2) qui peut être utilisée pour localiser le sous-ensemble de données. L'URL de base est le chemin complet pour l'entrée suivi par une barre oblique ("/"). La valeur "." indique qu'un sous-ensemble de données est situé directement en dessous de celui-ci. Plusieurs valeurs indiquent des copies dupliquées du sous-ensemble de données.

Par exemple, si l'ensemble de données "/folder/site/" a une entrée "public-folder" avec un attribut de sous-ensemble de données de ".", il existe alors un ensemble de données "/folder/site/public-folder/". Si au lieu de cela la valeur de l'attribut de sous-ensemble de données était "//other.acap.domain//folder/site/public-folder/", cela indiquerait que l'ensemble de données est en fait situé sur un serveur ACAP différent.

Un ensemble de données peut être créé en mémorisant un attribut "subdataset" qui comporte ".", et une sous-hiérarchie d'ensembles de données est supprimée en mémorisant une valeur NIL pour l'attribut "subdataset" dans l'entrée pour l'ensemble de données parent.

Cet attribut a une validation de syntaxe forcée. Précisément, si il est fait une tentative pour mémoriser avec STORE une valeur qui n'est pas dans la liste (autre que NIL), une liste vide, ou si une des valeurs ne suit pas les règles de syntaxe des URL [RFC1738], [RRF1808], il en résultera alors une réponse NO avec un code de réponse INVALID.

3.1.2 Métadonnées d'attribut

Chaque attribut est constitué d'éléments de métadonnées qui le décrivent, de sa valeur et de tous contrôles d'accès associés. Les éléments de métadonnées peuvent être en lecture seule, auquel cas il n'est jamais permis au client de modifier l'élément, ou en lecture-écriture, auquel cas le client peut modifier l'élément si la liste de contrôle d'accès (ACL, *access control list*) le permet.

Les éléments de métadonnées suivants sont définis dans la présente spécification :

acl
C'est la liste de contrôle d'accès pour l'attribut, s'il en existe une. Si l'attribut n'a pas d'ACL, NIL est retourné. En lecture-écriture. Voir au paragraphe 3.5 le contenu d'une ACL.

attribut
C'est le nom de l'attribut. En lecture seule.

myrights
C'est l'ensemble des droits que le client a sur l'attribut. En lecture seule. Voir au paragraphe 3.5 les droits possibles.

taille
C'est la longueur de la valeur. Dans le cas d'une valeur multiple, c'est une liste de la longueur de chaque valeur. En lecture seule.

valeur
C'est la valeur. Pour une valeur multiple, c'est une liste de valeurs seules. En lecture écriture.

Des éléments supplémentaires de métadonnées peuvent être définis dans des extensions à ce protocole. Les serveurs DOIVENT répondre aux métadonnées non reconnues en retournant un résultat d'achèvement de commande BAD.

3.2 Schéma d'URL ACAP

Les URL ACAP sont utilisés au sein du protocole ACAP pour l'attribut "sous-ensemble de données", pour les renvois documentaires et pour les héritages. Ils fournissent une syntaxe pratique pour se référer aux autres ensembles de données ACAP. L'URL ACAP suit la syntaxe commune de schéma Internet définie dans la [RFC1738] sauf que les mots de passe en clair ne sont pas permis. Si :<accès> est omis, l'accès par défaut est 674.

Un URL ACAP a la forme générale suivante :

url-acap = "acap://" url-server "/" url-enc-entry [url-filter] [url-extension]

L'élément <url-server> comporte le nom d'hôte, et un nom d'utilisateur facultatif, le mécanisme d'authentification et un numéro d'accès. L'élément <url-enc-entry> contient le nom d'un chemin d'entrée codé conformément aux règles de la [RFC1738].

L'élément <url-filter> est une liste facultative de noms d'attributs intéressants. S'il est omis, l'URL se réfère à tous les attributs de l'entrée désignée. L'élément <url-extension> est réservé aux extensions à ce schéma d'URL.

Noter que les caractères non sûrs ou réservés tels que " " ou "?" DOIVENT être codés en hexadécimal comme décrit dans la spécification des URL [RFC1738]. Les octets codés en hexadécimal sont interprétés conformément à UTF-8 [RFC2044].

3.2.1 Nom d'utilisateur et mécanisme d'authentification d'URL ACAP

Un nom d'utilisateur et/ou un mécanisme d'authentification peuvent être fournis. Ils sont utilisés dans la commande "AUTHENTICATE" après l'établissement de la connexion avec le serveur ACAP. Si un nom d'utilisateur ou un mécanisme d'authentification n'est pas fourni, le mécanisme SASL ANONYMOUS [RFC2245] est alors utilisé par défaut. Si un mécanisme d'authentification est fourni sans nom d'utilisateur, il DEVRAIT alors en être obtenu un du mécanisme spécifié ou en être demandé un à l'utilisateur, selon ce qui est approprié. Si un nom d'utilisateur est fourni sans un mécanisme d'authentification, ";AUTH=*" est alors supposé.

Le paramètre d'authentification ";AUTH=" est interprété comme décrit dans le schéma d'URL IMAP [RFC2192].

Noter que si un caractère non sûr ou réservé tel que " " ou ";" est présent dans le nom d'utilisateur ou le mécanisme d'authentification, il DOIT être codé comme décrit dans la spécification d'URL [RFC1738].

3.2.2 URL ACAP relatifs

Comme ACAP utilise "/" comme séparateur hiérarchique pour les chemins d'ensembles de données, cela fonctionne bien avec les règles des URL relatifs définis dans la spécification des URL relatifs [RFC1808].

L'élément grammatical <aauth> est considéré comme faisant partie du nom d'utilisateur pour les besoins de la résolution des URL ACAP relatifs.

L'URL de base pour un URL relatif mémorisé dans une valeur d'attribut est formé en prenant le chemin vers l'ensemble de données qui contient cet attribut, en ajoutant un "/" suivi par le nom d'entrée de l'entrée qui contient cet attribut suivi par"/".

3.3 Contextes

Un contexte est un sous-ensemble d'entrées dans un ou des ensembles de données, créés par une commande SEARCH avec un modificateur MAKECONTEXT. Les noms de contextes sont des chaînes générées par le client et elles ne doivent pas commencer par le caractère barre oblique ('/').

Lorsque un client crée un contexte, il peut demander une notification automatique des changements. Un client peut aussi demander l'énumération des entrées au sein d'un contexte. L'énumération simplifie la mise en œuvre d'une "barre de défilement virtuelle" par le client.

Un contexte n'existe qu'au sein de la session ACAP dans laquelle il a été créé. Lorsque la connexion est fermée, tous les contextes associés à cette connexion sont automatiquement éliminés. Un serveur est obligé de prendre en charge au moins 100 contextes actifs au sein d'une session. Si le serveur accepte une limite supérieure, il doit l'annoncer dans une capacité CONTEXTLIMIT.

3.4 Comparateurs

Un comparateur est une fonction désignée qui prend deux valeurs en entrée et peut être utilisée pour effectuer une ou plusieurs des quatre opérations de comparaison suivantes : rangement, égalité, correspondance de préfixe et de sous chaîne.

L'opération de rangement est utilisée aussi bien pour le modificateur de recherche SORT que pour les clés de recherche COMPARE et COMPARESTRICT. Les comparateurs de rangement peuvent déterminer la préséance ordinale de toute paire de valeurs. Lorsque il est utilisé pour le rangement, un nom de comparateur peut être muni du préfixe "+" ou "-" pour indiquer que le rangement devrait être respectivement en ordre normal ou en ordre inverse. Si aucun préfixe n'est inclus, "+" est supposé.

Pour les besoins d'un rangement, un comparateur peut désigner certaines valeurs comme ayant une préséance ordinale indéfinie. De telles valeurs sont toujours collationnées avec une valeur égale après toutes les autres valeurs, sans considération de l'ordre normal ou inverse qui est utilisé. Sauf si la définition du comparateur spécifie autre chose, les valeurs multiples et les valeurs NIL ont une préséance ordinale indéfinie.

L'opération d'égalité est utilisée pour le modificateur de recherche EQUAL, et détermine simplement si les deux valeurs sont considérées comme égales par la fonction de comparateur. Lors de la comparaison d'une seule valeur à une valeur multiple, les deux sont considérées comme égales si une des valeurs de la valeur multiple est égale à la valeur seule.

L'opération correspondance de préfixe est utilisée pour le modificateur de recherche PREFIX, et détermine simplement si la valeur de recherche est un préfixe de l'élément objet de la recherche. Dans le cas de recherche de préfixe sur une valeur multiple, la correspondance est réussie si la valeur est un préfixe de l'une des valeurs multiples.

L'opération de correspondance de sous chaîne est utilisée pour le modificateur de recherche SUBSTRING, et détermine simplement si une valeur de recherche est une sous chaîne de l'élément objet de la recherche. Dans le cas d'une recherche de sous chaîne sur une valeur multiple, la correspondance est réussie si la valeur est une sous chaîne de l'une des valeurs multiples.

Les règles de désignation et d'enregistrement des comparateurs seront définies dans une future spécification. Les serveurs DOIVENT répondre aux comparateurs inconnus ou utilisés à tort par un résultat d'achèvement de commande BAD.

Les comparateurs suivants sont définis par la présente norme et DOIVENT être mis en œuvre :

i;octet
Opérations : rangement, égalité, correspondance de préfixe, correspondance de sous chaîne

Pour la collation, le comparateur i;octet interprète la valeur d'un attribut comme une série d'octets non signés avec des valeurs ordinales de 0 à 255. Lors du rangement de deux chaînes, chaque paire d'octets est comparée en séquence jusqu'à ce que les octets soient inégaux ou que la fin de la chaîne soit atteinte. Lorsque on collationne deux chaînes où la plus courte est un préfixe de la plus longue, la plus courte chaîne est interprétée comme ayant une plus petite valeur ordinale. Les formes "i;octet" ou "+i;octet" collationnent d'abord les plus petites valeurs ordinales, et la forme "-i;octet" collationne d'abord les plus grandes valeurs ordinales.

Pour la fonction d'égalité, deux chaînes sont égales si elles ont la même longueur et contiennent les mêmes octets dans le même ordre. NIL n'est égal qu'à lui-même.

Pour les valeurs non binaires, non nulles, le rangement par i;octet est équivalent à la fonction strcmp() de l'ANSI C [ISO-C] appliquée à des représentations de chaînes C des valeurs. Pour des valeurs non binaires, non nulles, une correspondance de sous-chaîne i;octet est équivalente à la fonction ANSI C strstr() appliquée aux représentations de chaîne C des valeurs.

i;ascii-casemap
Opérations : rangement, égalité, correspondance de préfixe, correspondance de sous-chaîne.

Le comparateur i;ascii-casemap applique d'abord aux valeurs d'attribut une transposition qui traduit toutes les lettres US-ASCII en majuscules (les valeurs d'octets de 0x61 à 0x7A sont respectivement traduites en valeurs d'octet de 0x41 à 0x5A), puis elle applique le comparateur i;octet comme décrit ci-dessus. Avec cette fonction, les valeurs "hello" et "HELLO" ont la même valeur ordinale et sont considérées comme égales.

i;ascii-numeric
Opérations : rangement, égalité

Le comparateur i;ascii-numeric interprète les chaînes comme des entiers décimaux positifs représentés comme des chiffres US-ASCII. Toutes les valeurs qui ne commencent pas par un chiffre US-ASCII sont considérées comme égales à une valeur ordinale supérieure à tous les attributs non NUL de valeur unique. Autrement, tous les chiffres US-ASCII (les valeurs d'octet de 0x30 à 0x39) sont interprétées en commençant au début de la chaîne jusqu'au premier caractère qui n'est pas un chiffre ou à la fin de la chaîne.

3.5 Listes de contrôle d'accès

Une liste de contrôle d'accès (ACL, *Access Control List*) est un ensemble d'identifiants, de paires de droits utilisées pour interdire l'accès à un certain ensemble de données, et d'attributs, au sein d'une entrée. Une ACL est représentée par une valeur multiple dont chaque valeur contient un identifiant suivi par un caractère de tabulation suivi par les droits. La syntaxe est définie par la règle "acl" dans la syntaxe formelle de la section 8.

L'identifiant est une chaîne UTF-8. L'identifiant "anyone" *(tout le monde)* est réservé pour se référer à l'identité universelle (toutes les authentifications, y compris les anonymes). Toutes les chaînes de nom d'utilisateur acceptées par la commande AUTHENTICATE pour s'authentifier auprès du serveur ACAP sont réservées comme identifiants pour l'utilisateur correspondant. Les identifiants qui commencent par un caractère barre oblique ("/") sont réservés pour les groupes d'autorisation qui seront définis dans une future spécification. Les identifiants PEUVENT être préfixés d'un trait d'union ("-") pour indiquer une révocation des droits. Tous les autres identifiants ont des significations définies par la mise en œuvre.

Les droits sont une chaîne qui énumère un ensemble (éventuellement vide) de caractères alphanumériques, chaque caractère énumérant un ensemble d'opérations qui sont l'objet d'un contrôle. Les lettres sont réservées pour les droits "standard", énumérés ci-dessous. L'ensemble des droits standard ne peut être étendu que par une RFC en cours de normalisation ou expérimentale approuvée par l'IESG. Les droits sont réservé aux droits définis par la mise en œuvre ou le site. Les droits standard actuellement définis sont :

x - recherche (utilise la clé de recherche EQUAL avec le comparateur i;octet)
r - lecture (accès avec la commande SEARCH)
w - écriture (modifie avec la commande STORE)
i - insérer (effectue STORE sur une valeur précédemment NIL)
a - administrer (effectue SETACL ou STORE sur l'attribut/métadonnée d'ACL)

Une mise en œuvre peut forcer les droits à être toujours ou jamais accordés. En particulier, les mises en œuvre sont supposées accorder les droits implicites de lecture et administrer à la mémorisation de l'ensemble de données personnel de l'usager afin d'éviter des problèmes de déni de service. Les droits ne sont jamais liés, à la différence de l'extension ACL de IMAP [RFC2086].

Il est possible que plusieurs identifiants dans une liste de contrôle d'accès s'appliquent à un certain usager (ou autre identité d'authentification). Par exemple, une ACL peut inclure des droits accordés à l'identifiant qui correspond à l'usager, un ou plusieurs identifiants définis par la mise en œuvre correspondant à des groupes qui comportent l'usager, et/ou l'identifiant "tout le monde". Ces droits sont combinés en prenant l'union de tous les droits positifs qui s'appliquent à un certain usager et en soustrayant l'union de tous les droits négatifs qui s'appliquent à cet usager. Un client PEUT éviter ce calcul en utilisant la commande MYRIGHTS et les éléments de métadonnées.

Chaque attribut de chaque entrée d'un ensemble de données peut éventuellement avoir une ACL. Si un attribut dans une entrée n'a pas d'ACL, l'accès est alors contrôlé par une ACL par défaut pour cet attribut dans l'ensemble de données, si il existe. Si il n'y a pas d'ACL par défaut pour cet attribut dans l'ensemble de données, l'accès est contrôlé par une ACL par défaut pour cet ensemble de données. L'ACL par défaut pour un ensemble de données doit exister.

Pour effectuer un accès ou une manipulation sur une entrée dans un ensemble de données, le client doit avoir des droits 'r' sur l'attribut "entry" de l'entrée. Les mises en œuvre devraient faire attention à ne pas révéler via des messages d'erreur l'existence d'une entrée pour laquelle le client n'a pas les droits 'r'. Un client n'a pas besoin d'accéder à l'attribut "subdataset" de l'ensemble de données parent pour accéder au contenu d'un ensemble de données.

Beaucoup des commandes et réponses d'une ACL comportent un paramètre "acl object", pour spécifier à quoi s'applique l'ACL. C'est une liste entre parenthèses. La liste contient juste le nom de l'ensemble de données lorsque elle se réfère à l'ACL par défaut pour un ensemble de données. La liste contient un nom d'ensemble de données et un nom d'attribut lorsque elle se réfère à l'ACL par défaut pour un attribut dans un ensemble de données. La liste contient un nom d'ensemble de données, un nom d'attribut, et un nom d'entrée lorsque elle se réfère à l'ACL pour un attribut dans une entrée dans un ensemble de données.

3.6 Codes de réponse du serveur

Une réponse OK, NO, BAD, ALERT ou BYE de la part du serveur PEUT contenir un code de réponse pour décrire l'événement d'une façon plus détaillée analysable par la machine. Un code de réponse consiste en données entre parenthèses sous la forme d'un atome, éventuellement suivi par une espace et des arguments. Les codes de réponse sont définis lorsque il y a une action spécifique que peut faire un client sur la base des informations supplémentaires. Afin de prendre en charge les extensions futures, le code de réponse est représenté comme une hiérarchie dont les éléments sont séparés par une barre oblique, chaque niveau de hiérarchie représentant un détail plus précis de l'erreur. Les clients DOIVENT tolérer les détails hiérarchiques supplémentaires de code de réponse qu'ils ne comprennent pas.

Les codes de réponse actuellement définis sont :

AUTH-TOO-WEAK *(authentification trop faible)*
Ce code de réponse est retourné sur un résultat NO étiqueté provenant d'une commande AUTHENTICATE. Il indique que la politique de sécurité du site interdit l'utilisation du mécanisme demandé pour l'identité d'authentification spécifiée.

ENCRYPT-NEEDED *(chiffrement nécessaire)*
Ce code de réponse est retourné sur un résultat NO étiqueté provenant d'une commande AUTHENTICATE. Il indique que la politique de sécurité de ce site exige l'utilisation d'un mécanisme de chiffrement fort pour l'identité et le mécanisme d'authentification spécifiés.

INVALID
Ce code de réponse indique qu'une commande STORE comportait des données que la mise en œuvre de serveur ne permet pas. Il NE DOIT PAS être utilisé sauf si la spécification de classe d'ensemble de données pour l'attribut en question permet explicitement la validation forcée du serveur. L'argument est l'attribut qui était invalide.

MODIFIED
Ce code de réponse indique qu'une mémorisation conditionnelle a échoué parce que l'heure de modification sur l'entrée est plus tardive que l'heure de modification spécifiée avec le modificateur UNCHANGEDSINCE de la commande STORE. L'argument est l'entrée qui a été modifiée.

NOEXIST
Ce code de réponse indique qu'une recherche ou une mémorisation NOCREATE a échoué parce que un ensemble de données spécifié n'existe pas. L'argument est l'ensemble de données qui n'existe pas.

PERMISSION
Une commande a échoué à cause d'une permission insuffisante sur la base de la liste de contrôle d'accès ou de droits implicites. L'argument est l'objet acl-object qui a causé l'échec de la permission.

QUOTA
Une commande STORE ou SETACL qui aurait augmenté la taille de l'ensemble de données a échoué à cause d'un quota insuffisant.

REFER
Ce code de réponse peut être retourné dans une réponse NO étiquetée à toute commande qui prend un nom d'ensemble de données comme paramètre. Il a un ou plusieurs arguments avec la syntaxe des URL relatifs. C'est un référant, qui indique que la commande devrait être restituée en utilisant un des URL relatifs.

SASL
Ce code de réponse peut survenir dans la réponse OK étiquetée sur une commande AUTHENTICATE réussie et il inclut les données de réponse finale facultative du serveur provenant du serveur comme spécifié par SASL [RFC2222].

TOOMANY *(trop nombreuses)*
Ce code de réponse peut être retourné dans une réponse OK étiquetée à la commande SEARCH qui comporte le modificateur LIMIT. L'argument retourne le nombre total d'entrées qui correspondent.

TOOOLD *(trop vieux)*
L'heure de modification spécifiée dans la commande DELETEDSINCE est trop ancienne, de sorte que les informations deletedsince *(supprimé depuis)* ne sont plus disponibles.

TRANSITION-NEEDED *(transition nécessaire)*
Ce code de réponse survient sur une réponse NO à une commande AUTHENTICATE. Il indique que le nom de l'usager est

valide, mais l'entrée dans la base de données d'authentification doit être mise à jour afin de permettre l'authentification avec le mécanisme spécifié. Cela peut arriver si un usager a une entrée dans une base de données d'authentification de système comme Unix /etc/passwd, mais n'a pas les accréditifs convenables pour l'utilisation par le mécanisme spécifié.

TRYLATER *(essayer plus tard)*
Une commande a échoué à cause d'une défaillance temporaire du serveur. Le client PEUT continuer à utiliser les informations locales et essayer la commande plus tard.

TRYFREECONTEXT *(essayer un contexte libre)*
Ce code de réponse peut être retourné dans une réponse NO étiquetée à une commande SEARCH qui comporte le modificateur MAKECONTEXT. Il indique qu'un nouveau contexte ne peut pas être créé à cause de la limite du serveur sur le nombre de contextes existants.

WAYTOOMANY
Ce code de réponse peut être retourné dans une réponse NO étiquetée à une commande SEARCH qui comporte un modificateur de recherche HARDLIMIT.Il indique que la commande SEARCH aurait retourné plus d'entrées que permis par HARDLIMIT.

Les codes de réponse supplémentaires DEVRONT être enregistrés auprès de l'IANA conformément aux procédures du paragraphe 7.2. Les mises en œuvre de client DOIVEN tolérer les codes de réponse qu'elles ne reconnaissent pas.

4. Conventions de l'espace de noms

4.1 Espace de noms Dataset

L'espace de noms dataset est une hiérarchie séparée par des barres obliques. Le premier composant de l'espace de noms dataset est une classe d'ensemble de données. Les classes d'ensembles de données DOIVENT avoir un préfixe de fabricant (fabricant.<fabricant/produit>) ou être spécifiées dans une RFC en cours de normalisation ou une RFC expérimentale approuvée par l'IESG. Voir au paragraphe 7.3 le gabarit d'enregistrement.

Le second composant du nom d'ensemble de données est "site", "groupe", "hôte", ou "usager" qui se réfère à des données respectivement au niveau du serveur, du groupe administratif, de l'hôte, et de l'usager.

Pour les domaines "groupe", "hôte", et "usager", le troisième composant du chemin est le nom du groupe, le nom de domaine pleinement qualifié de l'hôte, ou le nom de l'utilisateur. Un chemin de la forme "/<classe-d'ensemble-de-données>/~/" est une abréviation pratique de "/<classe-d'ensemble-de-données>/usager/<usager-actuel>/".

Les noms d'ensembles de données qui commencent par "/titulaire/" sont réservés comme solutions de remplacement de l'espace de noms. Cela donne un moyen pour voir toutes les classes d'ensembles de données qu'utilise un titulaire particulier. Par exemple, "/titulaire/~/<classe-d'ensemble-de-données>/" est un nom de remplacement pour "/<classe-d'ensemble-de-données>/~/". Le champ par titulaire donne un moyen pour voir une liste de classes d'ensembles de données possédées par un certain usager ; cela se fait en utilisant l'ensemble de données "/titulaire/usager/<usager-actuel>/" avec le modificateur NOINHERIT SEARCH.

L'ensemble de données "/" peut être utilisé pour trouver toutes les classes d'ensembles de données visibles à l'usager actuel. Un ensemble de données de la forme "/<classe-d'ensemble-de-données>/usager/" peut être utilisé pour trouver tous les usagers qui ont fait un ensemble de données ou une entrée de cette classe visible à l'usager en cours.

La syntaxe formelle d'un nom d'ensemble de données est défini par la règle "nom-d'ensemble-de-données" au paragraphe 4.3.

4.2 Espace de noms d'attribut

Les noms d'attribut qui ne contiennent pas de point (".") sont réservés aux attributs normalisés qui ont une signification dans tous les ensembles de données. Afin de simplifier les mises en œuvre de client, l'espace de noms d'attribut est destiné à être unique sur l'ensemble des ensembles de données. Pour ce faire, les noms d'attribut sont munis d'un préfixe avec le nom de classe d'ensemble de données suivi d'un point (".") Les attributs qui affectent la gestion de l'ensemble de données sont munis d'un préfixe avec "ensemble-de-données.". De plus, un sous arbre de l'espace de noms d'attribut "fabricant." peut être enregistré auprès de l'IANA conformément aux règles du paragraphe 7.4. Les mises en œuvre de ACAP sont invitées à aider à définir des spécifications interopérables de classes d'ensembles de données plutôt que d'utiliser l'espace de noms d'attribut privé.

Certains usagers ou sites pourraient souhaiter ajouter leurs propres attributs privés à certaines classes d'ensembles de données. Pour le permettre, les sous-arbres "usager.<nom-d'usager>." et "site." de l'espace de nom d'attribut sont réservés aux attributs respectivement spécifiques de l'usager et spécifiques du site, et ne seront pas normalisés. De tels attributs ne sont pas interopérables et sont donc déconseillés en faveur de la définition d'attributs standard. Une future extension est prévue pour permettre la découverte de la syntaxe des attributs spécifiques de l'usager ou du site. Les clients qui souhaitent prendre en charge l'affichage des attributs spécifiques de l'usager ou du site devraient afficher la valeur de tout attribut "usager.<nom-d'usager>." ou "site." d'une seule valeur non NIL qui a une syntaxe UTF-8 valide.

La syntaxe formelle d'un nom d'attribut est définie par la règle "nom-d'attribut" dans le paragraphe suivant.

4.3 Syntaxe formelle pour les espaces de noms d'ensemble de données et d'attribut

Les conventions de dénomination pour les ensembles de données et les attributs sont définies par l'ABNF qui suit. Noter que cette grammaire ne fait pas partie de la syntaxe du protocole ACAP dans la section 8, car les noms d'ensembles de données et d'attributs sont codés comme des chaînes au sein du protocole ACAP.

```
attribute-dacl  = "dataset.acl" *("." name-component)

attribute-dset  = dataset-std 1*("." name-component)     ;; DOIT être défini dans une spécification de classe d'ensemble de
                                                            données

attribute-name  = attribute-std / attr-site / attr-user / vendor-name

attribute-std   = "entry" / "subdataset" / "modtime" / "dataset.inherit" / attribute-dacl / attribute-dset

attr-site       = "site" 1*("." name-component)

attr-user       = "user." name-component 1*("." name-component)

byowner         = "/byowner/" owner "/" [dataset-class "/" dataset-sub]

dataset-class   = dataset-std / vendor-name

dataset-normal  = "/" [dataset-class "/" (owner-prefix / dataset-tail)]

dataset-name    = byowner / dataset-normal

dataset-std     = name-component          ;; DOIT être enregistré auprès de l'IANA et la spécification DOIT être publiée comme
                                             RFC en cours de normalisation ou expérimentale approuvée par l'IESG.

dataset-sub     = *(dname-component "/")  ;; Les règles pour cette portion de l'espace de nom peuvent être restreintes par la
                                             spécification de classe d'ensemble de données.

dataset-tail    = owner "/" dataset-sub

dname-component = 1*UTF8-CHAR             ;; NE DOIT PAS commencer par "." ou contenir "/".

name-component  = 1*UTF8-CHAR             ;; NE DOIT PAS contenir ".", "/", "%", ou "*".

owner           = "site" / owner-host / owner-group / owner-user / "~"

owner-group     = "group/" dname-component

owner-host      = "host/" dname-component

owner-prefix    = "group/" / "host/" / "user/"

owner-user      = "user/" dname-component

vendor-name     = vendor-token *("." name-component)
```

vendor-token = "vendor." name-component ;; DOIT être enregistré auprès de l'IANA.

5. Gestion de l'ensemble de données

L'entrée avec un nom vide ("") dans l'ensemble de données est utilisée pour contenir les informations de gestion de l'ensemble de données considéré comme un tout.

5.1 Héritage de l'ensemble de données

Il est possible à un ensemble de données d'hériter des données d'un autre. L'ensemble de données duquel les données sont héritées est appelé l'ensemble de données de base. Les données dans l'ensemble de données de base apparaissent dans l'ensemble de données héritier, sauf lorsque elles sont subrogées par un STORE sur l'ensemble de données héritier.

L'ensemble de données de base est habituellement un ensemble par défaut à l'échelle d'un système ou d'un groupe. Un ensemble de données à l'échelle d'un système a normalement un ensemble de données héritier par usager, ce qui permet à chaque usager de l'ajouter ou de modifier les données par défaut, selon ce qui est approprié.

Une entrée qui existe dans les deux ensembles de données héritier et de base hérite d'une heure de modification égale à la plus grande des deux. Un attribut dans une telle entrée est hérité de l'ensemble de données de base si il n'a jamais été modifié par une commande STORE dans l'ensemble de données héritier ou si DEFAULT était accolé à cet attribut. Cela permet aux entrées par défaut d'être amendées plutôt que remplacées dans l'ensemble de données héritier.

L'attribut "subdataset" n'est pas hérité directement. Si l'ensemble de données de base comporte un attribut "subdataset" et si l'ensemble de données héritier n'en comporte pas, l'attribut "subdataset" va alors hériter d'une valeur virtuelle d'une liste contenant un ".". Le sous-ensemble de données à ce nœud est dit être un ensemble de données "virtuel" car il est simplement une copie virtuelle de l'ensemble de données approprié avec tous les attributs "sous-ensemble de données" changés en une liste contenant un ".". Un ensemble de données virtuel n'est pas visible si NOINHERIT est spécifié dans la commande SEARCH.

Les serveurs DOIVENT prendre en charge au moins deux niveaux d'héritage. Cela permet à un ensemble de données d'un usager tel que "/options/usager/fred/commun" d'hériter d'un ensemble de données de groupe tel que "/options/groupe/opérateurs dinosaures/commun" qui à son tour hérite d'un ensemble de données à l'échelle du serveur tel que "/options/site/commun".

5.2 Attributs de l'ensemble de données

Les attributs suivants s'appliquent à la gestion de l'ensemble de données lorsque ils sont mémorisés dans l'entrée "" de l'ensemble de données. Ces attributs ne sont pas hérités.

dataset.acl
Cela contient la liste de contrôle d'accès par défaut pour l'ensemble de données. Cet attribut est validé, de sorte qu'une liste de contrôle d'accès invalide dans une commande STORE résultera en une réponse NO avec un code de réponse INVALIDE.

dataset.acl.<attribut>
Cela contient la liste de contrôle d'accès par défaut pour un attribut au sein de l'ensemble de données. Cet attribut est validé, de sorte qu'une liste de contrôle d'accès invalide dans une commande STORE résultera en une réponse NO avec un code de réponse INVALIDE.

dataset.inherit
Cela contient le nom d'un ensemble de données duquel hériter conformément aux règles du paragraphe précédent. Cet attribut PEUT se référer à un ensemble de données non existant, auquel cas rien n'est hérité. Cet attribut est validé de sorte qu'une syntaxe illégale d'ensemble de données ou une tentative de mémoriser une valeur multiple résultera en une réponse NO avec un code de réponse INVALIDE.

5.3 Création de l'ensemble de données

Lorsque un ensemble de données est créé pour la première fois (en mémorisant un "." dans le sous-attribut d'ensemble de données ou en mémorisant une entrée dans un ensemble de données précédemment non existant) les attributs d'ensemble de données sont initialisés avec les valeurs provenant de l'ensemble de données parent dans la hiérarchie "/byowner/". Dans le cas d'un attribut "dataset.inherit", on ajoute le composant de hiérarchie appropriée. Par exemple, avec l'entrée suivante (noter que \t se réfère au caractère US-ASCII tabulation horizontale) :

chemin d'entrée "/byowner/user/joe/"
dataset.acl ("joe\txrwia" "fred\txr")
dataset.inherit "/byowner/site"

Si une nouvelle classe d'ensemble de données "/byowner/user/joe/new" est créée, elle va avoir les attributs d'ensemble de données suivants :

chemin d'entrée "/byowner/user/joe/new/"
dataset.acl ("joe\txrwia" "fred\txr")
dataset.inherit "/byowner/site/new"

Noter que l'ensemble de données "/byowner/user/joe/new/" est équivalent à "/new/user/joe/".

5.4 Capacités des classes d'ensembles de données

Certaines classes d'ensembles de données ou caractéristiques de classe d'ensemble de données peuvent n'être utiles que si il y a une prise en charge d'un client de mise à jour ou de serveur intégré active pour la caractéristique. La classe d'ensemble de données "capability" est réservée pour permettre aux clients ou serveurs d'annoncer de telles caractéristiques. L'attribut "entry" au sein de cette classe d'ensemble de données est le nom de la classe d'ensemble de données dont les caractéristiques sont décrites. Les attributs sont préfixés de "capability.<classe-d'ensemble-de-données>." et sont définis par la spécification de classe d'ensemble de données appropriée.

Comme il est possible à un usager non privilégié de faire fonctionner un client actif pour lui-même, un ensemble de données de capacités par usager est utile. L'ensemble de données "/capability/~/" contient des informations sur toutes les caractéristiques disponibles à l'usager (via l'héritage) et l'ensemble de données "/capability/site/" contient les informations sur toutes les caractéristiques prises en charge par le site.

5.5 Quotas d'ensembles de données

La gestion et la portée des quotas dépendent de la mise en œuvre. Les clients peuvent vérifier la limite et l'usage du quota applicable (en octets) avec la commande GETQUOTA. Les serveurs peuvent notifier au client une situation de faible quota avec la réponse non étiquetée QUOTA.

6. Spécifications des commandes et des réponses

Les commandes et réponses ACAP sont décrites dans la présente section. Les commandes sont organisées d'abord par l'état dans lequel elles sont permises, puis par une catégorie générale de type de commande.

Les arguments de commande, identifiés par "Arguments:" dans les descriptions de commande ci-dessous, sont décrits par fonction, non par syntaxe. La syntaxe précise des arguments de commande est décrite dans la section 8 sur la syntaxe formelle.

Certaines commandes causent le retour de données spécifiques du serveur ; elles sont identifiées par "Données :" dans la descriptions de commande ci-dessous. Voir la descriptions de réponse dans la section Réponses pour des informations sur ces réponses, et dans la section 8 "Syntaxe formelle" pour la syntaxe précise de ces réponses. Il est possible aux données de serveur d'être transmises comme résultat de toute commande ; donc, les commandes qui n'exigent pas spécifiquement de données de serveur spécifient "pas de données spécifiques pour cette commande" au lieu de "aucune".

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

6.1 Connexion initiale

Au démarrage d'une session, le serveur envoie une des deux réponses non étiquetées : ACAP ou BYE. La réponse BYE non étiquetée est décrite au paragraphe 6.2.8.

6.1.1 Réponse ACAP non étiquetée

Données : liste de capacités

La réponse ACAP non étiquetée indique que la session est prête pour accepter des commandes et contenir une liste de capacités séparées par des espaces que prend en charge le serveur. Chaque capacité est représentée par une liste contenant le nom de la capacité suivi facultativement par des arguments de chaîne spécifiques de la capacité.

Les noms de capacité ACAP DOIVENT être enregistrés auprès de l'IANA conformément aux règles du paragraphe 7.1.

Les mises en œuvre de client NE DEVRAIENPAS exiger de nom de capacité au delà de ceux définis dans la présente spécification, et DOIVENT tolérer tous les noms de capacité inconnus. Une mise en œuvre de client PEUT être configurée de façon à exiger des mécanismes SASL autres que CRAM-MD5 [RFC2195] pour des raisons de politique de sécurité du site.

Les capacités initiales suivantes sont définies :

CONTEXTLIMIT
La capacité CONTEXTLIMIT a un argument qui est un nombre décrivant le nombre maximum de contextes que le serveur prend en charge par connexion. Le nombre 0 indique que le serveur n'a pas de limite, autrement, ce nombre DOIT être supérieur à 100.

IMPLEMENTATION
La capacité IMPLEMENTATION a un argument qui est une chaîne décrivant la mise en œuvre de serveur. Les clients ACAP NE DOIVENT PAS modifier leur comportement sur la base de cette valeur. Elle est principalement destinée à des fins de débogage.

SASL
La capacité SASL comporte une liste des mécanismes d'authentification acceptés par le serveur. Voir au paragraphe 6.3.1.

Exemple :
S: * ACAP (IMPLEMENTATION "ACME v3.5") (SASL "CRAM-MD5") (CONTEXTLIMIT "200")

6.2 Dans tous les états

Les commandes et réponses suivantes sont valides dans tous les états.

6.2.1 Commande NOOP

Argument : aucun

Données ; pas de données spécifiques pour cette commande (mais voir ci-dessous).

Résultat : OK - noop achevé
BAD - commande inconnue ou arguments invalide

La commande NOOP réussit toujours. Elle ne fait rien. Elle peut être utilisée pour remettre à zéro tout temporisateur d'auto déconnexion sur inactivité sur le serveur.

Exemple :
C: a002 NOOP
S: a002 OK "NOOP completed"

6.2.2 Commande LANG

Arguments : liste de préférences de langage
Données : réponse intermédiaire : LANG
Résultat : OK - langage terminé
NO – pas de langage correspondant disponible
BAD - commande inconnue ou arguments invalides

Un ou plusieurs arguments sont fournis pour indiquer les langages préférés du client [RFC1766] pour les messages d'erreur. Le serveur va confronter chaque préférence du client dans l'ordre à son tableau interne des langues de chaîne d'erreur disponibles. Pour que la préférence du client corresponde à un langage du serveur, l'étiquette de langage du client DOIT être un préfixe de l'étiquette du serveur et correspondre jusqu'à un "-" ou la fin de la chaîne. Si une correspondance est trouvée, le serveur retourne une réponse LANG intermédiaire et une réponse OK. La réponse LANG indique le langage réel choisi et les comparateurs appropriés à utiliser avec les langages énumérés dans la commande LANG.

Si aucune commande LANG n'est produite, toutes les chaînes de texte d'erreur DOIVENT être dans le langage enregistré "i-default" [RFC2277], destiné à une audience internationale.

Exemple :
C: A003 LANG "fr-ca" "fr" "en-ca" "en-uk"
S: A003 LANG "fr-ca" "i;octet" "i;ascii-numeric" "i;ascii-casemap" "en;primary" "fr;primary"
S: A003 OK "Bonjour"

6.2.3 Réponse intermédiaire LANG

Données : langage pour les comparateurs appropriés de réponse d'erreur.

La réponse LANG indique le langage qui sera utilisé pour les réponses d'erreur et les comparateurs qui sont appropriés pour les langages énumérés dans la commande LANG. Les comparateurs DEVRAIENT être dans l'ordre approximatif du plus efficace (normalement "i;octet") au plus approprié pour le texte lisible par l'homme dans la langue préferrée.

6.2.4 Commande LOGOUT

Arguments : aucun
Données : réponse non étiquetée obligatoire : BYE
Résultat : OK – déconnexion achevée
BAD - commande inconnue ou arguments invalides

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

Exemple :
C: A023 LOGOUT
S: * BYE "Le serveur ACAP se déconnecte"
S: A023 OK "LOGOUT terminé " (Serveur et client closent alors la connexion)

6.2.5 Réponse OK

Données : code de réponse facultatif en texte lisible par l'homme.

La réponse OK indique un message d'information de la part du serveur. Lorsque il est étiqueté, il indique la réussite de l'achèvement de la commande associée. Le texte lisible par l'homme peut être présenté à l'usager comme un message d'information. La forme non étiquetée indique un message de pure information ; la nature des informations PEUT être indiquée par un code de réponse.

Exemple :
S: * OK "Le serveur maître ACAP est rétabli"

6.2.6 Réponse NO

Données : code de réponse facultatif en texte lisible par l'homme.

La réponse NO indique un message d'erreur de fonctionnement de la part du serveur. Lorsque il est étiqueté, il indique l'achèvement non réussi de la commande associée. La forme non étiquetée indique un avertissement ; la commande peut encore se terminer avec succès. Le texte lisible par l'homme décrit la condition.

Exemple :
C: A010 SEARCH "/addressbook/" DEPTH 3 RETURN ("*") EQUAL "entry" "+i;octet" "bozo"
S: * NO "Le serveur maître ACAP est en panne, vos données peuvent être périmées."
S: A010 OK "recherche effectuée"
...
C: A222 STORE ("/folder/site/comp.mail.misc" "folder.creation-time" "19951206103412")
S: A222 NO (PERMISSION ("/folder/site/")) "Permission refusée"

6.2.7 Réponse BAD

Données : code de réponse facultatif lisible par l'homme.

La réponse BAD indique un message d'erreur de la part du serveur. Lorsque elle est étiquetée, elle fait rapport d'une erreur de niveau protocole dans la commande du client ; l'étiquette indique la commande qui a causé l'erreur. La forme non étiquetée indique une erreur de niveau protocole pour laquelle la commande associée ne peut pas être déterminée ; elle peut aussi indiquer une défaillance interne du serveur. Le texte lisible par l'homme décrit la condition.

Exemple :
C: ...ligne vide...
S: * BAD "Ligne de commande vide"
C: A443 BLURDYBLOOP
S: A443 BAD "commande inconnue"
C: A444 NOOP Hello
S: A444 BAD "arguments invalides"

6.2.8 Réponse BYE non étiquetée

Données : code de réponse facultatif en texte lisible par l'homme.

La réponse BYE non étiquetée indique que le serveur est sur le point de clore la connexion. Le texte lisible par l'homme peut être affiché à l'usager dans un rapport d'état par le client. La réponse BYE peut être envoyée au titre d'une séquence de déconnexion normale, ou au titre d'une annonce de clôture d'urgence par le serveur. Elle est aussi utilisée par certaines mises en œuvre de serveur comme annonce d'une auto déconnexion sur inactivité.

Cette réponse est aussi utilisée comme un des deux accueils possibles au démarrage de session. Elle indique que le serveur ne veut pas accepter une session de la part de ce client.

Exemple :
S: * BYE "Auto déconnexion ; trop longtemps inactif."

6.2.9 Réponse ALERT non étiquetée

Données : code de réponse facultatif en texte lisible par l'homme.

Le texte lisible par l'homme contient un message spécial généré par l'homme qui DOIT être présenté à l'usager d'une façon qui attire l'attention de l'usager sur le message. Il est destiné à être utilisé pour des messages vitaux de la part de l'administrateur du serveur à l'usager, tels qu'un avertissement que le serveur va bientôt être fermé pour sa maintenance.

Exemple :
S: * ALERT "Ce serveur ACAP va fermer dans 10 minutes pour la maintenance du système."

6.3 État non authentifié

Dans l'état non authentifié, la commande AUTHENTICATE établit l'authentification et entre dans l'état authentifié. La commande AUTHENTICATE fournit un mécanisme général pour diverses techniques d'authentification.

Les mises en œuvre de serveur peuvent permettre un accès non authentifié à certaines informations en prenant en charge les mécanisme SASL ANONYMOUS de la [RFC2245].

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

Seules les commandes tous états (NOOP, LANG et LOGOUT) et la commande AUTHENTICATE sont valides dans l'état non authentifié.

6.3.1 Commande AUTHENTICATE

Arguments : réponse initiale facultative de nom de mécanisme SASL
Données : des données de continuation peuvent être exigées
Résultat : OK – authentification achevée, maintenant dans l'état authentifié
NO – échec de l'authentification : mécanisme d'authentification non pris en charge, accréditifs rejetés
BAD - commande inconnue ou arguments invalides, échange d'authentification annulé

La commande AUTHENTICATE indique au serveur un mécanisme d'authentification SASL [RFC2222]. Si le serveur prend en charge le mécanisme d'authentification requis, il effectue un échange de protocole d'authentification et identifie l'usager. Facultativement, il négocie aussi une couche de sécurité pour les interactions de protocole suivantes. Si le mécanisme d'authentification requis n'est pas accepté, le serveur rejette la commande AUTHENTICATE en envoyant une réponse NO étiquetée.

L'échange de protocole d'authentification consiste en une série de défis du serveur et de réponses du client qui sont spécifiques du mécanisme d'authentification. Un défi du serveur consiste en une demande de continuation de commande avec le jeton "+" suivi d'une chaîne. La réponse du client consiste en une ligne constituée d'une chaîne. Si le client souhaite annuler un échange d'authentification, il devrait produire une ligne avec une seule "*" non étiquetée. Si le serveur reçoit une telle réponse, il doit rejeter la commande AUTHENTICATE en envoyant une réponse BAD étiquetée.

L'argument facultatif de réponse initiale à la commande AUTHENTICATE est utilisé pour faire l'économie d'un aller-retour lorsque on utilise des mécanismes d'authentification qui sont définis pour n'envoyer aucune donnée dans le défi initial. Lorsque l'argument réponse initiale est utilisé avec un tel mécanisme, le défi vide initial n'est pas envoyé au client et le serveur utilise les données dans l'argument réponse initiale comme si elles étaient envoyées en réponse au défi vide. Si l'argument réponse initiale à la commande AUTHENTICATE est utilisé avec un mécanisme qui envoie des données dans le défi initial, le serveur rejette la commande AUTHENTICATE en envoyant une réponse NO étiquetée.

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

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

Toutes les mises en œuvre ACAP DOIVENT appliquer le mécanisme SASL CRAM-MD5 de la [RFC2195], bien qu'elles PUISSENT offrir une option de configuration pour le désactiver si la politique de sécurité du site l'impose. L'exemple ci-dessous est le même que celui décrit dans la spécification CRAM-MD5.

Si une commande AUTHENTICATE échoue avec une réponse NO, le client peut essayer un autre mécanisme d'authentification en produisant une autre commande AUTHENTICATE. En d'autres termes, le client peut demander les types d'authentification en ordre de préférence décroissant.

Exemple :
S: * ACAP (IMPLEMENTATION "Blorfysoft v3.5") (SASL "CRAM-MD5" "KERBEROS_V4")
C: A001 AUTHENTICATE "CRAM-MD5"
S: + "<1896.697170952@postoffice.reston.mci.net>"
C: "tim b913a602c7eda7a495b4e6e7334d3890"
S: A001 OK "Authentification CRAM-MD5 réussie"

6.4 Recherche

Cette section décrit la commande SEARCH, pour restituer des données à partir des ensembles de données.

6.4.1 Commande SEARCH

Arguments : liste facultative d'ensembles de données ou de contextes de critères de recherche de modificateurs
Données : réponse intermédiaires : ENTRY, MODTIME, REFER
réponses non étiquetées : ADDTO, REMOVEFROM, CHANGE, MODTIME
Résultat : OK - recherche terminée
NO – échec de la recherche : la recherche ne peut être effectuée
BAD - commande inconnue ou arguments invalides

La commande SEARCH identifie un sous ensemble d'entrées dans un ensemble de données et retourne des informations sur ce sous ensemble au client. Les entrées et attributs hérités sont inclus dans la recherche sauf si le modificateur de recherche NOINHERIT est inclus ou si l'usager n'a pas la permission de lire les attributs dans l'ensemble de données de base.

Le premier argument de SEARCH identifie ce qui doit être recherché. Si la chaîne commence par une barre oblique ("/"), c'est le nom d'un ensemble de données à rechercher, autrement, c'est un nom d'un contexte qui a été créé par une commande SEARCH donnée précédemment dans la session.

Une commande SEARCH réussie PEUT résulter en des réponses ENTRY intermédiaires et DOIT résulter en une réponse intermédiaire MODTIME.

À la suite de cela sont zéro, un ou plusieurs modificateurs à la recherche. Chaque modificateur peut être spécifié au plus une fois. Les modificateurs définis sont :

numéro de DEPTH *(profondeur)*
La commande SEARCH va traverser l'arbre des ensembles de données jusqu'à la profondeur spécifiée. Les réponses ENTRY vont inclure le chemin complet jusqu'à l'entrée. Une valeur de "0" indique que la recherche devrait traverser l'arbre entier. Une valeur de "1" est la valeur par défaut et indique seulement que l'ensemble de données spécifié devrait être recherché. Si un ensemble de données est traversé et qu'il n'est pas localisé sur le serveur actuel, une réponse intermédiaire REFER est alors retournée pour ce sous-arbre et la recherche continue.

nombre de HARDLIMIT *(limite matérielle)*
Si la commande SEARCH devait résulter en plus que le nombre d'entrées, la commande SEARCH échoue avec un résultat d'achèvement NO avec un code de réponse WAYTOOMANY.

nombre de numéros LIMIT
Limite le nombre de réponses ENTRY intermédiaires que peut générer la recherche. Le premier argument numérique spécifie la limite ; le second nombre spécifie le nombre d'entrées à retourner si le nombre de correspondances dépasse la limite. Si la limite est dépassée, la commande SEARCH réussit quand même, et retourne le nombre total de correspondances dans un code de réponse TOOMANY dans la réponse OK étiquetée.

MAKECONTEXT [ENUMERATE] [NOTIFY] contexte
Amène la commande SEARCH à créer un contexte avec le nom donné dans l'argument pour se référer aux entrées correspondantes. Si la commande SEARCH réussit, le nom du contexte peut alors être donné comme argument aux commandes SEARCH suivantes pour rechercher l'ensemble des entrées correspondantes. Si un contexte avec le nom spécifié existe déjà, il est d'abord libéré. Si un nouveau contexte ne peut pas être créé du fait de la limite du serveur sur le nombre de contextes existants, la commande échoue, et retourne un code de réponse TRYFREECONTEXT dans la réponse d'achèvement NO.

Les arguments facultatifs "ENUMERATE" et "NOTIFY" peuvent être inclus pour demander l'énumération du contexte (pour les barres de déroulement virtuelles) ou des notifications de changement pour le contexte. Si "NOTIFY" n'est pas demandé, le contexte représente une photographie des entrées au moment de la production de la commande SEARCH.

ENUMERATE demande que le contenu du contexte soit ordonné conformément au modificateur SORT et que des numéros de séquence, commençant à un, soient alloués aux entrées dans le contexte. Cela permet au modificateur RANGE d'être utilisé pour aller chercher des portions du contexte ordonné.

NOTIFY demande que le serveur envoie des réponses ADDTO, REMOVEFROM, CHANGE, et MODTIME non étiquetées alors que le contexte créé par cette commande SEARCH existe. Le serveur PEUT produire des notifications ADDTO, REMOVEFROM, CHANGE et MODTIME non étiquetées pour un contexte à tout moment entre la production de la commande SEARCH avec MAKECONTEXT NOTIFY et l'achèvement d'une commande FREECONTEXT pour le contexte. Les notifications ne sont produites que pour les changements qui surviennent après que le serveur a reçu la commande SEARCH qui a créé le contexte. Après la production d'une séquence de notifications ADDTO, REMOVEFROM ou CHANGE, le serveur DOIT produire une notification MODTIME non étiquetée indiquant que le client a toutes les mises à jour

aux entrées dans le contexte jusque et y compris la valeur de modtime donnée. Les serveurs sont autorisés à un délai raisonnable pour grouper les notifications de changement avant de les envoyer au client.

Les arguments de position des notifications ADDTO, REMOVEFROM et CHANGE sont 0 si ENUMERATE n'est pas demandé.

NOINHERIT
Cela amène la commande SEARCH à fonctionner sans héritage. Cela peut être utilisé pour dire quelles valeurs sont des recouvrements explicites. Si MAKECONTEXT est aussi spécifié, le contexte créé est aussi non affecté par l'héritage.

RETURN (métadonnées...)
Spécifie ce qui doit être retourné dans les réponses ENTRY intermédiaires. Si ce modificateur n'est pas spécifié, aucune réponse ENTRY intermédiaire n'est retournée.

Entre les parenthèses se trouve une liste facultative d'attributs, suivis facultativement chacun par une liste entre parenthèses de métadonnées. Si la liste entre parenthèse de métadonnées n'est pas spécifiée, elle est de "(valeur)" par défaut.

Un nom d'attribut avec une "*" en queue demande que tous les attributs aient ce préfixe. Une "*" demande par elle-même tous les attributs. Si la liste entre parenthèses de métadonnées n'est pas spécifiée pour un attribut avec une "*" en queue, elle prend "(valeur d'attribut)" par défaut. Les résultats qui correspondent à un tel gabarit d'attribut sont groupés entre des parenthèses.

À la suite de la dernière réponse ENTRY intermédiaire, le serveur retourne une seule réponse MODTIME intermédiaire.

SORT (comparateur d'attribut ...)
Spécifie l'ordre dans lequel toutes les réponses ENTRY résultantes sont à retourner au client. Le modificateur SORT prend comme argument une liste entre parenthèses d'une ou plusieurs paires attribut/comparateur. L'attribut fait la liste des attributs à trier ; comparateur spécifie le nom de la règle de collationnement à appliquer aux valeurs de l'attribut. Les paires successives attribut/comparateur ne sont utilisées pour ordonner deux entrées que lorsque toutes les paires précédentes indiquent que les deux entrées collationnent la même valeur.

Si le modificateur SORT est utilisé en conjonction avec le modificateur MAKECONTEXT, le modificateur SORT spécifie l'ordre des entrées dans le contexte créé.

Si aucun modificateur SORT n'est spécifié, ou si aucune des paires attribut/comparateur n'indique un ordre pour les deux entrées, le serveur utilise l'ordre des entrées qui existe dans le contexte ou l'ensemble de données qui fait l'objet de la recherche.

Le critère de recherche suit les modificateurs. Les critères de recherche consistent en une ou plusieurs clés de recherche. Les clés de recherche peuvent être combinées en utilisant les clés de recherche AND, et OR. Par exemple, le critère :

AND COMPARE "modtime" "+i;octet" "19951206103400" COMPARE "modtime" "-i;octet" "19960112000000"

se réfère à toutes les entrées modifiées entre 10:34 le 6 décembre 1995 et minuit le 12 janvier 1996, en UTC.

Les clés de recherche actuellement définies sont les suivantes :

ALL : Cela correspond à toutes les entrées.

AND clé de recherche1 clé de recherche2 : Ce sont les entrées qui correspondent aux deux clés de recherche.

COMPARE valeur de comparateur d'attribut : Ce sont les entrées pour lesquelles la valeur de l'attribut spécifié se collationne en utilisant le comparateur spécifié de la même façon que la valeur spécifiée ou après elle.

COMPARESTRICT valeur de comparateur d'attribut : Ce sont les entrées pour lesquelles l'attribut spécifié se collationne en utilisant le comparateur spécifié après la valeur spécifiée.

EQUAL valeur de comparateur d'attribut : Ce sont les entrées pour lesquelles la valeur de l'attribut est égale à la valeur spécifiée en utilisant le comparateur spécifié.

NOT clé de recherche : Ce sont les entrées qui ne correspondent pas à la clé de recherche spécifiée.

OR clé de recherche1 clé de recherche2 : Ce sont les entrées qui correspondent à l'une ou l'autre clé de recherche.

PREFIX valeur de comparateur d'attribut : Ce sont les entrées qui commencent par la valeur spécifiée en utilisant le comparateur spécifié. Si le comparateur spécifié ne prend pas en charge la confrontation de sous-chaînes, une réponse BAD est retournée.

RANGE *(gamme)* heure de début et de fin : Ce sont les entrées qui sont dans la gamme spécifiée de l'ordre du contexte énuméré. L'entrée de plus faible rang dans le contexte reçoit le numéro un, la plus faible entrée suivante reçoit le numéro deux, et ainsi de suite. Les arguments numériques spécifient les plus faibles et plus forts numéros à atteindre. L'heure spécifie que le client a traité les notifications pour le contexte jusqu'à l'heure spécifiée. Si le contexte a été modifié depuis, le serveur DOIT soit retourner un NO avec un code de réponse MODIFIED, soit retourner les résultats que SEARCH aurait retourné si aucun des changements n'avait été fait depuis ce moment.
RANGE n'est permis que sur les contextes énumérés. Si RANGE est utilisé avec un ensemble de données ou un contexte non énuméré, le serveur DOIT retourner une réponse BAD.

SUBSTRING valeur de comparateur d'attribut : Ce sont les entrées qui contiennent la valeur spécifiée, en utilisant le comparateur spécifié. Si le comparateur spécifié ne prend pas en charge la confrontation de sous-chaînes, une réponse BAD est retournée.

6.4.2 Réponse intermédiaire ENTRY

Données : nom de l'entrée
données de l'entrée

La réponse intermédiaire ENTRY survient comme résultat d'une commande SEARCH ou STORE. C'est le moyen par lequel les entrées d'ensemble de données sont retournées au client.

La réponse ENTRY commence par le nom de l'entrée, si une commande SEARCH sans le modificateur DEPTH a été produite, ou par le chemin de l'entrée dans les autres cas. Ceci est suivi par un ensemble de zéro, un ou plusieurs éléments, un pour chaque élément de métadonnées dans le modificateur de recherche RETURN. Les résultats qui correspondent à un schéma d'attribut ou qui retournent plusieurs éléments de métadonnées sont groupés entre parenthèses.

6.4.3 Réponse intermédiaire MODTIME

Données : valeur de l'heure de modification

La réponse intermédiaire MODTIME survient comme résultat d'une commande SEARCH. Elle indique que le contexte qui vient juste d'être créé ou que les réponses ENTRY précédemment retournées incluent toutes les mises à jour à l'entrée retournée jusque et y compris la valeur de l'heure de modification dans l'argument.

6.4.4 Réponse intermédiaire REFER

Données : chemin de l'ensemble de données
URL relatifs ACAP

La réponse intermédiaire REFER survient comme résultat d'une commande SEARCH sur plusieurs niveaux où un des niveaux est localisé sur un serveur différent. La réponse indique l'ensemble de données qui n'est pas localisé sur le serveur en cours et un ou plusieurs URL relatifs ACAP pour indiquer où cet ensemble de données peut être trouvé.

6.4.5 Exemples de recherches

Voici quelques échanges de commandes SEARCH entre le client et le serveur :

```
C: A046 SEARCH "/addressbook/" DEPTH 3 RETURN ("addressbook.Alias" "addressbook.Email" "addressbook.List")
      OR NOT EQUAL "addressbook.Email" "i;octet" NIL NOT EQUAL "addressbook.List" "i;octet" NIL
S: A046 ENTRY "/addressbook/user/joe/A0345" "fred" "fred@stone.org" NIL
S: A046 ENTRY "/addressbook/user/fred/A0537" "joe" "joe@stone.org" NIL
S: A046 ENTRY "/addressbook/group/Dinosaur Operators/A423" "saurians" NIL "1"
S: A046 MODTIME "19970728105252"
S: A046 OK "SEARCH completed"
C: A047 SEARCH "/addressbook/user/fred/" RETURN ("*") EQUAL "entry" "i;octet" "A0345"
S: A047 ENTRY "A0345" (("modtime" "19970728102226")
```

("addressbook.Alias" "fred") ("addressbook.Email" "fred@stone.org") ("addressbook.CommonName" "Fred Flintstone") ("addressbook.Surname" "Flintstone") ("addressbook.GivenName" "Fred"))
S: A047 MODTIME "19970728105258"
S: A047 OK "SEARCH completed"
C: A048 SEARCH "/options/~/vendor.example/" RETURN ("option.value"("size" "value" "myrights")) SORT ("entry" "i;octet") COMPARE "modtime" "i;octet" "19970727123225"
S: A048 ENTRY "blurdybloop" (5 "ghoti" "rwia")
S: A048 ENTRY "buckybits" (2 "10" "rwia")
S: A048 ENTRY "windowSize" (7 "100x100" "rwia")
S: A048 MODTIME "19970728105304"
S: A048 OK "SEARCH completed"
C: A049 SEARCH "/addressbook/~/public" RETURN ("addressbook.Alias" "addressbook.Email") MAKECONTEXT ENUMERATE "blob" LIMIT 100 1 SORT ("addressbook.Alias" "i;octet") NOT EQUAL "addressbook.Email" NIL
S: A049 ENTRY "A437" "aaguy" "aaguy@stone.org"
S: A049 MODTIME "19970728105308"
S: A049 OK (TOOMANY 347) "Context 'blob' created"
C: A050 SEARCH "blob" RANGE 2 2 "19970728105308" ALL
S: A050 ENTRY "A238" "abguy" "abguy@stone.org"
S: A050 MODTIME "19970728105310"
S: A050 OK "SEARCH Completed"

6.5 Contextes

Les commandes suivantes utilisent des contextes créés par une commande SEARCH avec un modificateur MAKECONTEXT.

6.5.1 Commande FREECONTEXT

Arguments : nom de contexte
Données : pas de données spécifiques pour cette commande
Résultat : OK - freecontext terminé
NO – échec de freecontext : pas de tel contexte
BAD - commande inconnue ou arguments invalides

La commande FREECONTEXT amène le serveur à libérer tous les états associés au contexte désigné. Le contexte ne peut plus être recherché et le serveur ne va plus produire de réponses non étiquetées pour le contexte. Le contexte n'est plus compté dans la limite du nombre de contextes du serveur.

Exemple :
C: A683 FREECONTEXT "blurdybloop"
S: A683 OK "Freecontext completed"

6.5.2 Commande UPDATECONTEXT

Arguments : liste de noms de contextes
Données : réponses non étiquetées : ADDTO REMOVEFROM CHANGE MODTIME
Résultat : OK - Updatecontext terminé : toutes les mises à jour sont terminées
NO – échec de Updatecontext : aucun contexte tel ; ce n'est pas un contexte notifié
BAD - commande inconnue ou arguments invalides

La commande UPDATECONTEXT amène le serveur à s'assurer que le client est notifié de tous les changements connus du serveur pour les contextes listés comme arguments jusqu'à l'heure actuelle. Les contextes énumérés dans les arguments doivent avoir été précédemment donnés à une commande SEARCH réussie avec un modificateur MAKECONTEXT NOTIFY. Une réponse MODTIME non étiquetée DOIT être retournée si des métadonnées en lecture-écriture ont changé dans le contexte depuis la dernière MODTIME pour ce contexte. Cela inclut les métadonnées qui ne sont pas énumérées dans le modificateur RETURN pour ce contexte.

Alors qu'un serveur peut produire des ADDTO, REMOVEFROM, CHANGE, et MODTIME non étiquetées à tout moment, la commande UPDATECONTEXT est utilisée pour "inciter" le serveur à envoyer toutes les notifications qu'il n'a pas encore envoyées.

La commande UPDATECONTEXT NE DEVRAIT PAS être utilisée pour interroger sur les mises à jour.

Exemple :
C: Z4S9 UPDATECONTEXT "blurdybloop" "blarfl"
S: Z4S9 OK "Le client a été notifié de tous les changements"

6.5.3 Réponse non étiquetée ADDTO

Données : nom de contexte
nom d'entrée
position
liste de métadonnées

La réponse ADDTO non étiquetée informe le client qu'une entrée a été ajoutée à un contexte. La réponse inclut le numéro de position de l'entrée ajoutée (la première entrée dans le contexte est numérotée 1) et les métadonnées contenues dans l'entrée qui correspond à la déclaration RETURN lors de la création du contexte.

Pour les contextes d'énumération, la réponse ADDTO ajoute implicitement un à la position de tous les membres du contexte qui avaient des numéros de position supérieurs ou égaux au numéro de position ADDTO. Pour des contextes qui ne sont pas d'énumération, le champ position est toujours 0.

Exemple :
S: * ADDTO "blurdybloop" "fred" 15 ("addressbook.Email" "fred@stone.org")

6.5.4 Réponse non étiquetée REMOVEFROM

Données : nom de contexte
nom d'entrée
ancienne position

La réponse non étiquetée REMOVEFROM informe le client qu'une entrée a été retirée d'un contexte. La réponse inclut le numéro de position qu'utilisait l'entrée retirée (la première entrée dans le contexte porte le numéro 1).

Pour les contextes d'énumération, la réponse REMOVEFROM soustrait implicitement un des numéros de position de tous les membres du contexte qui avaient des numéros de position supérieurs au numéro de position de REMOVEFROM. Pour les contextes qui ne sont pas d'énumération, le champ position est toujours 0.

Exemple :
S: * REMOVEFROM "blurdybloop" "fred" 15

6.5.5 Réponse non étiquetée CHANGE

Données : nom de contexte
nom d'entrée
ancienne position
nouvelle position
liste des métadonnées

La réponse non étiquetée CHANGE informe le client qu'une entrée dans un contexte a soit changé de position dans le contexte, soit a changé les valeurs d'un ou de plusieurs des attributs spécifiés dans le modificateur RETURN lors de la création du contexte.

La réponse comporte les numéros de position précédent et actuel de l'entrée (qui sont 0 si ENUMERATE n'était pas spécifié sur le contexte) et les métadonnées d'attribut demandées dans le modificateur RETURN lors de la création du contexte.

Pour les contextes d'énumération, la réponse CHANGE change implicitement les numéros de position de toutes les entrées qui avaient des numéros de position compris entre l'ancienne et la nouvelle position. Si l'ancienne position est inférieure à la nouvelle position, celle-ci est soustraite de toutes les entrées qui avaient des numéros de position dans cette gamme. Autrement, on ajoute un à toutes les entrées qui avaient des numéros de position dans cette gamme. Si l'ancienne position et la nouvelle position sont la même, aucun renumérotage de position implicite ne va survenir.

Les réponses CHANGE ne sont pas produites pour les entrées qui ont subi un changement de position implicite du fait d'une autre réponse ADDTO, REMOVEFROM ou CHANGE.

Exemple :
S: * CHANGE "blurdybloop" "fred" 15 10 ("addressbook.Email" "fred@stone.org")

6.5.6 Réponse non étiquetée MODTIME

Données : nom de contexte
valeur de l'heure de modification

La réponse non étiquetée MODTIME informe le client qu'il a reçu toutes les mises à jour aux entrées dans le contexte qui ont des valeurs d'heure de modification inférieures ou égales à la valeur de modtime dans l'argument.

Exemple :
S: * MODTIME mycontext "19970320162338"

6.6 Modification d'ensemble de données

Les commandes et réponses suivantes traitent des modifications d'ensembles de données.

6.6.1 Commande STORE

Arguments : liste de mémorisation des entrées
Données : réponse intermédiaires : ENTRY
Résultat : OK – mémorisation achevée
NO – échec de mémorisation : on ne peut mémoriser ce nom spécifié UNCHANGEDSINCE et les entrées changées
BAD - commande inconnue ou arguments invalides, syntaxe UTF-8 invalide dans le nom d'attribut

Crée, modifie, ou supprime les entrées désignées dans les ensembles de données nommés. Les valeurs des métadonnées non spécifiées dans la commande ne sont pas changées. Régler la métadonnée "valeur" d'un attribut à NIL retire cet attribut de l'entrée. Régler la "valeur" de l'attribut "entrée" à NIL retire cette entrée de l'ensemble de données et annule les héritages pour toute l'entrée. Régler la "valeur" de l'attribut "entrée" à DEFAULT retire cette entrée de l'ensemble de données héritier et ramène l'entrée et ses attributs aux valeurs héritées, s'il en est. Changer la valeur de l'attribut "entrée" renomme l'entrée.

Mémoriser DEFAULT pour la métadonnée "valeur" d'un attribut est équivalent à mémoriser NIL, sauf que l'héritage est activé pour cet attribut. Si une valeur non NIL est héritée, une réponse intermédiaire ENTRY est alors générée pour notifier ce changement au client. La réponse ENTRY inclut le chemin de l'entrée et les métadonnées nom et valeur d'attribut pour chaque attribut qui est revenu à un réglage hérité non NIL.

Mémoriser NIL à la métadonnée "valeur" d'un attribut PEUT être traité comme équivalent à mémoriser DEFAULT pour cette "valeur" si il y a une valeur NIL dans l'ensemble de données de base.

La commande STORE est suivie par une ou plusieurs listes de mémorisation d'entrées. Chaque liste de mémorisation d'entrées commence par un chemin d'entrée suivi par les modificateurs STORE, suivis par zéro, un ou plusieurs éléments de mémorisation d'attribut. Chaque élément de mémorisation d'attribut est constitué du nom d'attribut suivi par NIL (pour retirer la valeur de l'attribut) par DEFAULT (pour faire revenir l'élément à une valeur héritée) par une seule valeur (pour régler la valeur unique de l'attribut) ou par une liste d'éléments de métadonnées à modifier. Les modificateurs STORE suivants peuvent être spécifiés :

NOCREATE
Par défaut, le serveur DOIT créer tous les ensembles de données nécessaires pour mémoriser l'entrée, y compris plusieurs niveaux de hiérarchie. Si NOCREATE est spécifié, la commande STORE va échouer avec une erreur NOEXIST sauf si l'ensemble de données parent existe déjà.

UNCHANGEDSINCE
Si l' heure de modification "modtime" de l'entrée est plus tardive que l'heure "unchangedsince" *(non modifiée depuis)*, la mémorisation échoue alors avec un code de réponse MODIFIED. L'utilisation de UNCHANGEDSINCE avec une heure de

"00000101000000" va toujours échouer si l'entrée existe. Les clients qui écrivent à un ensemble de données partagé sont invités à utiliser UNCHANGEDSINCE lorsque ils modifient une entrée existante.

Le serveur DOIT soit faire tous les changements spécifié dans une seule commande STORE, soit n'en faire aucune. Si cela réussit, le serveur DOIT mettre à jour l'attribut "modtime" pour chaque entrée qui a été changée.

Il est illégal d'énumérer deux fois des éléments de métadonnées au sein d'un attribut, ni des attributs au sein d'une entrée ni un chemin d'entrée. Le serveur DOIT retourner une réponse BAD si cela arrive.

Le serveur PEUT réarranger les chaînes dans une valeur multiple sur STORE et PEUT retirer les chaînes dupliquées. Cependant, SEARCH DOIT retourner des valeurs multiples et les métadonnées de liste de taille associée dans un ordre cohérent.

Exemple :
C: A342 STORE ("/addressbook/user/fred/ABC547"
"addressbook.TelephoneNumber" "555-1234"
"addressbook.CommonName" "Barney Rubble"
"addressbook.AlternateNames" ("value" ("Barnacus Rubble" "Coco Puffs Thief"))
"addressbook.Email" NIL)
S: A342 OK "Store completed"
C: A343 STORE ("/addressbook/user/joe/ABD42" UNCHANGEDSINCE "19970320162338"
"user.joe.hair-length" "10 inches")
S: A343 NO (MODIFIED) "'ABD42' has been changed by somebody else."
C: A344 STORE ("/addressbook/group/Developers/ACD54" "entry" NIL)
S: A344 OK "Store completed"
C: A345 STORE ("/option/~/common/SMTPserver" "option.value" DEFAULT)
S: A345 ENTRY "/option/~/common/SMTPserver" "option.value" "smtp.server.do.main"
S: A345 OK "Store completed"
C: A347 STORE ("/addressbook/~/" "dataset.inherit" "/addressbook/group/Developers")
S: A347 OK "Store completed"

6.6.2 Commande DELETEDSINCE

Arguments : nom de l'ensemble de données
heure
Données : réponse intermédiaire: DELETED
Résultat : OK - DELETEDSINCE achevé
NO – échec de DELETEDSINCE : l'ensemble de données ne peut être lu ; la date est trop loin dans le passé
BAD - commande inconnue ou arguments invalides

La commande DELETEDSINCE retourne dans les réponses intermédiaires DELETED les noms des entrées qui ont été supprimées de l'ensemble de données désigné depuis l'heure donnée.

Les serveurs peuvent imposer une limite au nombre ou à l'âge des noms des entrées supprimées dont ils gardent trace. Si le serveur n'a pas le retour des informations à l'heure spécifiée, la commande échoue, retournant un code de réponse TOOOLD *(trop vieux)* dans la réponse NO.

Exemple :
C: Z4S9 DELETEDSINCE "/folder/site/" 19951205103412
S: Z4S9 DELETED "blurdybloop"
S: Z4S9 DELETED "anteaters"
S: Z4S9 OK "DELETEDSINCE completed"
C: Z4U3 DELETEDSINCE "/folder/site/" 19951009040854
S: Z4U3 NO (TOOOLD) "Je n'ai pas ces informations"

6.6.3 Réponse intermédiaire DELETED

Données : nom de l'entrée

La réponse intermédiaire DELETED survient en résultat d'une commande DELETEDSINCE. Elle retourne une entrée qui a été supprimée de l'ensemble de données spécifié dans la commande DELETEDSINCE.

6.7 Commandes de liste de contrôle d'accès

Les commande de cette section sont utilisées pour gérer les listes de contrôle d'accès.

6.7.1 Commande SETACL

Arguments : objet acl
identifiant d'authentification
droits d'accès
Données : pas de données spécifiques pour cette commande
Résultat : OK - setacl terminé
NO – échec de setacl : on ne peut pas établir d'acl
BAD - commande inconnue ou arguments invalides

La commande SETACL change la liste de contrôle d'accès sur l'objet spécifié de sorte que l'identifiant spécifié reçoive les permissions énumérées dans les droits. Si l'objet n'avait pas précédemment de liste de contrôle d'accès, il en est créé une.

Exemple :
C: A123 SETACL ("/addressbook/~/public/") "anyone" "r"
S: A123 OK "Setacl complete"
C: A124 SETACL ("/folder/site/") "B1FF" "rwa"
S: A124 NO (PERMISSION ("/folder/site/")) "'Il n'est pas permis à B1FF' de modifier les droits d'accès à '/folder/site/'"

6.7.2 Commande DELETEACL

Arguments : objet acl
identifiant d'authentification facultatif
Données : aucune données spécifiques pour cette commande
Résultat : OK - deleteacl terminé
NO – échec de deleteacl : on ne peut pas supprimer acl
BAD - commande inconnue ou arguments invalides

Si l'argument d'identifiant facultatif est donné, la commande DELETEACL retire toute portion de la liste de contrôle d'accès sur l'objet spécifié pour l'identifiant spécifié.

Si l'argument d'identifiant facultatif n'est pas donné, la commande DELETEACL retire entièrement la liste ACL de l'objet, causant le transfert du contrôle d'accès à une ACL par défaut de niveau supérieur. Cette forme de la commande DELETEACL n'est pas permise sur l'ACL par défaut pour un ensemble de données et les serveurs DOIVENT retourner un BAD.

Exemple :
C: A223 DELETEACL ("/addressbook/~/public") "anyone"
S: A223 OK "Deleteacl complete"
C: A224 DELETEACL ("/folder/site")
S: A224 BAD "Can't delete ACL from dataset"
C: A225 DELETEACL ("/addressbook/user/fred" "addressbook.Email" "barney")
S: A225 OK "Deleteacl complete"

6.7.3 Commande MYRIGHTS

Arguments : objet acl
Données : réponse intermédiaires: MYRIGHTS
Résultat : OK - myrights achevé
NO – échec de myrights : on ne peut obtenir les droits
BAD - commande inconnue ou arguments invalides

La commande MYRIGHTS retourne l'ensemble des droits que le client a sur l'ensemble de données ou attribut d'ensemble de données en question .

Exemple :
C: A003 MYRIGHTS ("/folder/site")
S: A003 MYRIGHTS "r"
S: A003 OK "Myrights complete"

6.7.4 Réponse intermédiaire MYRIGHTS

Données : droits

La réponse MYRIGHTS survient comme résultat d'une commande MYRIGHTS. L'argument est l'ensemble des droits que le client a sur l'objet auquel se réfère la commande MYRIGHTS.

6.7.5 Commande LISTRIGHTS

Arguments : objet acl
identifiant d'authentification
Données : réponses non étiquetées : LISTRIGHTS
Résultat : OK - listrights terminée
NO – échec de listrights : on ne peut pas obtenir la liste des droits
BAD - commande inconnue ou arguments invalides

La commande LISTRIGHTS prend un objet et un identifiant et retourne les informations sur les droits que l'utilisateur actuel peut révoquer ou accorder à cet identifiant dans l'ACL pour cet objet.

Exemple :
C: a001 LISTRIGHTS ("/folder/~/") "smith"
S: a001 LISTRIGHTS "xra" "w" "i"
S: a001 OK Listrights completed
C: a005 LISTRIGHTS ("/folder/site/archive/imap") "anyone"
S: a005 LISTRIGHTS "" "x" "r" "w" "i"
S: a005 OK Listrights completed

6.7.6 Réponse intermédiaire LISTRIGHTS

Données : droits requis
liste des droits facultatifs

La réponse LISTRIGHTS survient comme résultat d'une commande LISTRIGHTS. Le premier argument est une chaîne qui contient l'ensemble (éventuellement vide) de droits qui seront toujours accordés à l'identifiant sur l'ensemble de données ou l'attribut.

Après cela sont zéro, une ou plusieurs chaînes contenant chacune un seul droit que l'utilisateur actuel peut révoquer ou accorder à l'identifiant dans l'ensemble de données ou attribut.

Le même droit NE DOIT PAS figurer plus d'une fois dans la liste dans la réponse LISTRIGHTS.

6.8 Quotas

Cette section définit les commandes et réponses qui se rapportent aux quotas.

6.8.1 Commande GETQUOTA

Arguments : ensemble de données
Données : réponses non étiquetées : QUOTA
Résultat : OK – informations de quota retournées
NO – échec de Quota : on ne peut accéder à la limite de ressource ; pas de limite de ressource
BAD - commande inconnue ou arguments invalides

La commande GETQUOTA prend le nom d'un ensemble de données et retourne dans une réponse QUOTA non étiquetée le nom de l'ensemble de données, la limite du quota en octets qui s'applique à cet ensemble de données et l'usage du quota au sein de cette limite. La portée d'une limite de quota dépend de la mise en œuvre.

Exemple :
C: A043 GETQUOTA "/option/user/fred/common"
S: * QUOTA "/option/user/fred/common" 1048576 2475
S: A043 OK "Getquota completed"

6.8.3 Réponse non étiquetée QUOTA

Données : ensemble de données
limite de quota en octets
quantité de la limite de quota utilisée
données d'extension

La réponse QUOTA non étiquetée est générée comme résultat d'une commande GETQUOTA ou PEUT être générée par le serveur en réponse à une commande SEARCH ou STORE pour avertir de la forte utilisation d'un quota. Elle comporte le nom de l'ensemble de données applicable, la limite du quota en octets, l'usage du quota et des données d'extension facultatives. Les clients DOIVENT tolérer les données d'extension car leur usage est réservé pour une extension future .

6.9 Extensions

Afin de simplifier le processus d'extension du protocole, les clients DOIVENT tolérer les réponses de serveur inconnues qui satisfont à la syntaxe d'extension de réponse. De plus, les clients DOIVENT tolérer les codes de réponse de serveur inconnus qui satisfont à la syntaxe de resp-code-ext. La disponibilité de nouvelles commandes DOIT être annoncée via une capacité sur la ligne d'accueil initiale et de telles commandes DEVRAIENT satisfaire à la syntaxe d'extension de commande.

Les serveurs DOIVENT répondre aux commandes inconnues par un résultat d'achèvement de commande de BAD. Les serveurs DOIVENT sauter les littéraux non synchronisants qui sont contenus dans une commande inconnue. Cela peut être fait en supposant que la commande inconnue satisfait à la syntaxe d'extension de commande, ou en lisant une ligne à la fois et en vérifiant la syntaxe des littéraux non synchronisants à la fin de la ligne.

7. Procédures d'enregistrement

L'utilité d'ACAP vient de ce qu'il fournit un modèle de mémorisation structuré pour toutes sortes de données de configuration. Cependant, pour que son potentiel se réalise, il est important que la communauté de l'Internet s'efforce d'atteindre les objectifs suivants :

(1) Normalisation : Il est très important de normaliser les classes d'ensembles de données. Les auteurs souhaitent que ACAP atteigne le succès qu'à connu SNMP avec la définition des nombreuses MIB *(bases de données d'informations de gestion)* en cours de normalisation.

(2) Révision par les pairs : En l'absence de normalisation, il est important d'avoir la révision par les pairs d'une proposition d'amélioration de la qualité de son ingénierie. La révision par les pairs est fortement recommandée avant l'enregistrement. La liste de diffusion des mises en œuvre de ACAP à <ietf-acap@andrew.cmu.edu> devrait être utilisée à cette fin.

(3) Enregistrement : L'enregistrement sert à deux fins distinctes. Tout d'abord, il empêche l'utilisation du même nom pour un objet différent, et ensuite, il fournit une liste unifiée de ce qui peut être utilisé pour localiser les extensions ou classes d'ensembles de données existantes pour empêcher la duplication du travail

Les gabarits d'enregistrement suivants peuvent être utilisés pour enregistrer les éléments de protocole ACAP auprès de l'autorité d'allocation des numéros de l'Internet (IANA).

7.1 Capacités d'ACAP

Les nouvelles capacités ACAP DOIVENT être enregistrées avant leur utilisation. Une considération attentive devrait être portée avant toute extension du protocole aux problèmes de complexité ou d'interopérabilité. La révision des propositions sur la liste de diffusion des mises en œuvre de acap est vivement recommandées avant l'enregistrement.

À : iana@iana.org
Sujet : Enregistrement de capacité ACAP
Nom de la capacité :
Mot clé de la capacité :
Arguments de la capacité :
Spécifications publiées :
Adresse personnelle et de messagerie à contacter pour des informations complémentaires : (Facultatif mais vivement recommandé)

7.2 Codes de réponse ACAP

Les codes de réponse ACAP sont enregistrés au fil de leur dépôt. La révision des proposition sur la liste de diffusion des mises en œuvre de acap est vivement recommandée avant l'enregistrement.

À : iana@iana.org
Sujet : Enregistrement des codes de réponse ACAP
Code de réponse :
Arguments (utiliser l'ABNF pour spécifier la syntaxe) :
Objet :
Spécifications publiées :
Adresse personnelle et de messagerie à contacter pour des informations complémentaires : (Facultatif mais vivement recommandé)

7.3 Classes d'ensembles de données

Une classe d'ensemble de données fournit un ensemble d'attributs à utiliser dans une hiérarchie précise. Elle peut aussi définir des règles pour la hiérarchie d'ensemble de données au dessous de cette classe. Les spécifications de classes d'ensembles de données doivent être des RFC en cours de normalisation ou des RFC expérimentales approuvées par l'IESG.

À : iana@iana.org
Sujet : enregistrement de classe d'ensemble de données ACAP
Nom de classe d'ensemble de données/préfixe d'attribut :
Objet :
Spécifications publiées :
Adresse personnelle et de messagerie à contacter pour des informations complémentaires : (Facultatif mais vivement recommandé)

7.4 Sous arborescence de fabricant

Les fabricants peuvent réserver une portion de l'espace de noms ACAP pour un usage privé. Les noms de classes d'ensembles de données qui commencent par "fabricant.<nom_de_compagnie/produit>." sont réservé à l'utilisation par cette compagnie ou produit. De plus, tous les noms d'attribut qui commencent par "fabricant.<nom _de_compagnie/produit>." sont réservés à l'usage de cette compagnie ou produit une fois qu'ils ont été enregistrés. L'enregistrement est fait au fil des demandes. Chaque fois que possible, les classes d'attributs et d'ensembles de données privés devraient être évitées en faveur de l'amélioration de la définition de classes d'ensembles de données interopérables.

À : iana@iana.org
Sujet : Enregistrement de sous-arborescence de fabricant ACAP
Préfixe privé : fabricant.<nom_de_compagnie/produit>.
Adresse personnelle et de messagerie à contacter pour des informations complémentaires : (les noms et adresses des sociétés devraient être inclus lorsque c'est approprié)

8. Syntaxe formelle

La spécification de syntaxe qui suit utilise la notation en forme Backus-Naur augmenté (ABNF) telle que spécifiée dans la [RFC2234]. Cela utilise le cœur de règles de l'ABNF telles que spécifiées dans l'Appendice A de la spécification de l'ABNF [RFC2234].

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 n'a de signification que pour la clarté de l'exposé. Les mises en œuvre DOIVENT accepter ces chaînes de façon insensible à la casse.

La règle "d'acceuil-initial" ci-dessous définit l'accueil initial ACAP de la part du serveur. La règle "commande" ci-dessous définit la syntaxe pour les commandes envoyées par le client. La règle "réponse" ci-dessous définit la syntaxe des réponses envoyées par le serveur.

```
ATOM-CHAR    = "!" / %x23-27 / %x2A-5B / %x5D-7A / %x7C-7E  ;; Tout CHAR sauf ATOM-SPECIALS
```

```
ATOM-SPECIALS          = "(" / ")" / "{" / SP / CTL / QUOTED-SPECIALS

CHAR   = %x01-7F

DIGIT-NZ    = %x31-39                                   ;; chiffres différents de zéro ("1" - "9")

QUOTED-CHAR            = SAFE-UTF8-CHAR / "\" QUOTED-SPECIALS

QUOTED-SPECIALS    = <"> / "\"

SAFE-CHAR      = %x01-09 / %x0B-0C / %x0E-21 / %x23-5B / %x5D-7F
                                                       ;; tous TEXT-CHAR sauf QUOTED-SPECIALS

SAFE-UTF8-CHAR     = SAFE-CHAR / UTF8-2 / UTF8-3 / UTF8-4 / UTF8-5 / UTF8-6

TAG-CHAR       = %x21 / %x23-27 / %x2C-5B / %x5D-7A / %x7C-7E
                                                             ;; tous ATOM-CHAR sauf "*" ou "+"

TEXT-CHAR      = %x01-09 / %x0B-0C / %x0E-7F                ;; tous CHAR sauf CR et LF

TEXT-UTF8-CHAR         = SAFE-UTF8-CHAR / QUOTED-SPECIALS

UTF8-1 = %x80-BF

UTF8-2 = %xC0-DF UTF8-1

UTF8-3 = %xE0-EF 2UTF8-1

UTF8-4 = %xF0-F7 3UTF8-1

UTF8-5 = %xF8-FB 4UTF8-1

UTF8-6 = %xFC-FD 5UTF8-1

UTF8-CHAR      = TEXT-UTF8-CHAR / CR / LF

acl = "(" [acl-identrights *(SP acl-identrights)] ")" *(SPACE acl-identrights)] ")"

acl-identifier       = string-utf8       ;; NE DOIT PAS contain HTAB

acl-identrights                                = string-utf8          ;; L'identifiant suivi par une HTAB, suivie par les droits.

acl-delobject      = "(" ensemble de données SP attribut [SP nom_d'entrée] ")"

acl-object           = "(" ensemble de données SP attribut [SP nom_d'entrée]] ")"

acl-rights           = quoted

atom      = ALPHA *1023ATOM-CHAR

attribute              = string-utf8                  ;; le nom d'attribut séparé par des points NE DOIT PAS contenir "*" ou "%"

attribute-store      = attribut SP (value-nildef / "(" 1*(metadata-write-q SP value-store) ")")
                                                          ;; NE DOIT PAS contenir deux fois les mêmes métadonnées

auth-type            = <"> auth-type-name <">

auth-type-name    = iana-token                                ;; comme défini dans SASL [RFC2222]

commande    = étiquette SP (commande-any / commande-auth / commande-nonauth) CRLF     ;; en modal sur la base de l'état

commande-authent           = "AUTHENTICATE" SP auth-type [SP string] *(CRLF string)
```

```
commande-any   = "NOOP" / commande-lang / "LOGOUT" / commande-extend

commande-auth  = commande-delacl / commande-dsince / commande-freectx / commande-getquota /
                 commande-lrights / commande-myrights / commande-search / commande-setacl / commande-store
                                                        ;; n'est valide que dans l'état authentifié

commande-delacl          = "DELETEACL" SP acl-delobject [SP acl-identifier]

commande-dsince          = "DELETEDSINCE" SP ensemble de données SP heure

commande-extend          = extend-token [SP extension-data]

commande-freectx         = "FREECONTEXT" SP contexte

commande-getquota        = "GETQUOTA" SP ensemble de données

commande-lang  = "LANG" *(SP lang-tag)

commande-lrights         = "LISTRIGHTS" SP acl-object

commande-myrights        = "MYRIGHTS" SP acl-object

commande-nonauth         = commande-authent       ;; n'est valide que dans l'état authentifié

commande-search          = "SEARCH" SP (dataset / context) *(SP search-modifier) SP search-criteria
                                                  ;; NE DOIT PAS inclure deux fois le même modificateur de recherche

commande-setacl= "SETACL" SP acl-object SP acl-identifier SP acl-rights

commande-store  = "STORE" SP store-entry-list

comparator       = <"> comparator-name <">

comparator-name= ["+" / "-"] iana-token

context     = string-utf8                        ;; NE DOIT PAS commencer par une barre oblique ("/")

dataset     = string-utf8       ;; le nom d'ensemble de données séparé par barre oblique commence par une barre oblique

entry     = entry-name / entry-path

entry-name    = string-utf8       ;; entry name NE DOIT PAS contenir de barre oblique et NE DOIT PAS commencer par "."

entry-path         = string-utf8       ;; le chemin vers l'entrée séparé par barre oblique commence par une barre oblique

entry-relative      = string-utf8                   ;; chemin vers l'entrée éventuellement relatif

extend-token    = atom        ;; DOIT être défini par une RFC en cours de normaisation ou une extension de protocole
                              expérimentale approuvée par l'IESG

extension-data     = extension-item *(SP extension-item)

extension-item     = extend-token / chaîne / numéro / "(" [extension-data] ")"

iana-token          = atom                  ;; DOIT être enregistré auprès de l'IANA

initial-greeting    = "*" SP "ACAP" *(SP "(" init-capability ")") CRLF

init-capability     = init-cap-context / init-cap-extend / init-cap-implem / init-cap-sasl

init-cap-context   = "CONTEXTLIMIT" SP chaîne
```

init-cap-extend = iana-token [SP string-list]

init-cap-implem = "IMPLEMENTATION" SP chaîne

init-cap-sasl = "SASL" SP string-list

lang-tag = <"> Language-Tag <"> ;; La règle d'étiquette de langage est définie dans la [RFC1766]

littéral = "{" nombre ["+"] "}" CRLF *OCTET ;; Le nombre représente le nombre d'octets
;; DOIT être un littéral utf8 sauf pour les valeurs

littéral-utf8 = "{" nombre ["+"] "}" CRLF *UTF8-CHAR ;; Le nombre représente le nombre d'octets
;; et non le nombre de caractères

metadata = attribut ["(" metadata-type-list ")"] ;; attribut PEUT se terminer par "*" comme caractère générique

metadata-list = metadata *(SP metadata)

metadata-type = "attribut" / "myrights" / "taille" / "compte" / metadata-write

metadata-type-q = <"> metadata-type <">

metadata-type-list = metadata-type-q *(SP metadata-type-q)

metadata-write = "valeur" / "acl"

metadata-write-q = <"> metadata-write <">

nil = "NIL"

nombre = *CHIFFRE ;; Un nombre non signé de 32 bits . (0 <= n < 4 294 967 296)

nz-number = CHIFFRE-NZ *CHIFFRE ;; Un nombre non signé de 32 bits différent de zéro number.
;; (0 < n < 4 294 967 296)

position = nombre ;; "0" si le contexte n'est pas une énumération, autrement c'est fifférent de zéro

quota-limit = nombre

quota-usage = nombre

quoted = <"> *QUOTED-CHAR <"> ;; limité à 1024 octets entre les <">

réponse = réponse-addto / réponse-alert / réponse-bye / réponse-change / réponse-cont / réponse-deleted / réponse-done / réponse-entry / réponse-extend / réponse-listr / réponse-lang / réponse-mtimei / réponse-mtimeu / réponse-myright / réponse-quota / réponse-refer / réponse-remove / réponse-stat

réponse-addto = "*" SP "ADDTO" SP contexte SP nom_d'entrée SP position SP return-data-list

réponse-alert = "*" SP "ALERTE" SP corps_de_réponse CRLF ;; Le client DOIT afficher le texte de l'alerte à l'usager

réponse-bye = "*" SP "BYE" SP corps_de_réponse CRLF ;; Condition de déconnexion du serveur

réponse-change = "*" SP "CHANGE" SP contexte SP nom_d'entrée SP position SP position SP return-data-list

réponse-cont = "+" SP chaîne

réponse-deleted = étiquette SP "DELETED" SP nom_d'entrée

réponse-done = étiquette SP resp-cond-state CRLF

réponse-entry = étiquette SP "ENTRY" SP entry SP return-data-list

```
réponse-extend   = (étiquette / "*") SP extend-token [SP extension-data]

réponse-lang     = "*" SP "LANG" SP lang-tag 1*(SP comparator)

réponse-listr    = étiquette SP "LISTRIGHTS" SP acl-rights *(SP acl-rights)

réponse-mtimei   = étiquette SP "MODTIME" SP time

réponse-mtimeu   = "*" SP "MODTIME" SP context SP time

réponse-myright  = étiquette SP "MYRIGHTS" SP acl-rights

réponse-quota    = "*" SP "QUOTA" SP dataset SP quota-limit SP quota-usage [SP extension-data]

réponse-refer    = étiquette SP "REFER" SP dataset 1*(SP <"> url-relative <">)

réponse-remove   = "*" SP "REMOVEFROM" SP context SP entry-name SP position

réponse-stat     = "*" SP resp-cond-state CRLF

resp-body        = ["(" resp-code ")" SP] quoted

resp-code        = "AUTH-TOO-WEAK" / "ENCRYPT-NEEDED" / resp-code-inval / resp-code-mod / resp-code-noexist /
resp-code-perm / "QUOTA" / resp-code-refer / resp-code-sasl / resp-code-toomany / "TOOOLD" / "TRANSITION-NEEDED"
/ "TRYFREECONTEXT" / "TRYLATER" / "WAYTOOMANY" / resp-code-ext

resp-code-ext    = iana-token [SP extension-data]    ;; unknown codes DOIT be tolerated by the client

resp-code-inval  = "INVALID" 1*(SP entry-path SP attribute)

resp-code-mod    = "MODIFIED" SP entry-path

resp-code-noexist          = "NOEXIST" SP dataset

resp-code-perm   = "PERMISSION" SP acl-object

resp-code-refer  = "REFER" 1*(SP <"> url-relative <">)

resp-code-sasl   = "SASL" SP string

resp-code-toomany          = "TOOMANY" SP nz-number

resp-cond-state  = ("OK" / "NO" / "BAD") SP resp-body      ;; Status condition

return-attr-list     = "(" return-metalist *(SP return-metalist) ")"          ;; occurs when "*" in RETURN pattern on SEARCH

return-data      = return-metadata / return-metalist / return-attr-list

return-data-list     = return-data *(SP return-data)

return-metalist  = "(" return-metadata *(SP return-metadata) ")"       ;; occurs when multiple metadata items requested

return-metadata  = nil / string / value-list / acl

searchkey-equal  = "EQUAL" SP attribute SP comparator SP value-nil

searchkey-comp   = "COMPARE" SP attribute SP comparator SP value

searchkey-prefix = "PREFIX" SP attribute SP comparator SP value

searchkey-range  = "RANGE" SP nz-number SP nz-number SP time

searchkey-strict   = "COMPARESTRICT" SP attribute SP comparator SP value
```

```
searchkey-substr = "SUBSTRING" SP attribute SP comparator SP value

searchmod-depth = "DEPTH" SP number

searchmod-hard  = "HARDLIMIT" SP nz-number

searchmod-limit  = "LIMIT" SP number SP number

searchmod-make = "MAKECONTEXT" [SP "ENUMERATE"] [SP "NOTIFY"] SP context

searchmod-ninh  = "NOINHERIT"

searchmod-return= "RETURN" SP "(" [metadata-list] ")"

searchmod-sort   = "SORT" SP "(" sort-list ")"

search-criteria      = "ALL" / searchkey-equal / searchkey-comp / searchkey-strict / searchkey-range / searchkey-prefix /
                       searchkey-substr / "NOT" SP search-criteria / "OR" SP search-criteria SP search-criteria / "AND" SP search-
                       criteria SP search-criteria

search-modifier   = searchmod-depth / searchmod-hard / searchmod-limit / searchmod-make / searchmod-ninh / searchmod-
                       return / searchmod-sort

sort-list              = sort-item *(SP sort-item)

sort-item              = attribut SP comparateur

store-entry          = "(" entry-path *(SP store-modifier) *(SP attribute-store) ")"
                                                    ;; NE DOIT PAS inclure deux fois le même store-modifier
                                                    ;; NE DOIT PAS inclure deux fois le même attribut

store-entry-list     = store-entry *(SP store-entry)       ;; NE DOIT PAS inclure deux fois la même entrée

store-modifier     = store-mod-unchang / store-mod-nocreate

store-mod-nocreate          = "NOCREATE"

store-mod-unchang           = "UNCHANGEDSINCE" SP date

chaîne    = quoted / littéral

string-list            = chaîne *(SP chaîne)

string-utf8          = quoted / littéral-utf8

tag = 1*32TAG-CHAR

date          = <"> année mois jour heure minute seconde fraction_de_seconde <">              ;; Horodatage en UTC

jour              = 2CHIFFRES                    ;; 01-31

heure     = 2CHIFFRES    ;; 00-23

minute   = 2CHIFFRES    ;; 00-59

mois      = 2CHIFFRES    ;; 01-12

seconde           = 2CHIFFRES       ;; 00-60

fraction_de_seconde         = *CHIFFRES

année              = 4CHIFFRES
```

```
valeur          = chaîne

valeur-liste    = "(" [valeur *(SP valeur)] ")"

valeur-nil      = valeur / nil

valeur-nildef   = valeur-nil / "DEFAULT"

valeur-store    = valeur-nildef / valeur-liste / acl

url-acap = "acap://" url-serveur "/" url-enc-entrée [url-filtre] [url-extension]
                                                        ;; url-enc-entrée est interprété par rapport à "/"

url-attr-liste  = url-enc-attr *("&" url-enc-attr)

url-auth = ";AUTH=" ("*" / url-enc-auth)

url-achar   = uchar / "&" / "=" / "~"                   ;; Voir dans la RFC1738 la définition de "uchar"

url-char    = uchar / "=" / "~" / ":" / "@" / "/"       ;; Voir dans la RFC1738 la définition de "uchar"

url-enc-attr  = 1*url-char                      ;; version codée du nom d'attribut

url-enc-auth  = 1*url-achar                     ;; version codée du auth-type-name ci-dessus

url-enc-entry = 1*url-char                      ;; version codée de entry-relative ci-dessus

url-enc-user  = *url-achar                      ;; version codée de l'identifiant de connexion d'usager

url-extension   = *("?" 1*url-char)

url-filter = "?" url-attr-list

url-relative    = url-acap / [url-enc-entry] [url-filter]   ;; url-enc-entry se rapporte à l'URL de base

url-server      = [url-enc-user [url-auth] "@"] hostport    ;; Voir dans la RFC1738 la définition de "hostport"
```

9. Considérations sur l'utilisation multilingue

L'atelier sur les jeux de caractères de l'IAB [RFC2130] est arrivé à un certain nombre de conclusions qui ont influencé la conception deACAP. La décision d'utiliser l'UTF-8 comme schéma de codage des caractères se fonde sur ce travail. La commande LANG pour négocier un langage pour les messages d'erreur est aussi incluse.

Le paragraphe 3.4.5 du rapport de l'atelier sur les jeux de caractères de l'IAB déclare qu'il devrait y avoir un moyen d'identifier le langage naturel pour les chaînes lisibles par l'homme. Plusieurs propositions prometteuses ont été faites pour l'utiliser au sein d'ACAP, mais il n'y a pas eu de consensus clair sur une méthode unique à ce stade. Les règles suivantes devraient vraisemblablement permettre l'ajout de la prise en charge du multilingue à l'avenir :

(1) Un travail en cours appelé format de chaîne multilingue (MLSF, *Multi-Lingual String Format*) propose une couche par dessus UTF-8 qui utilise des séquences UTF-8 autrement illégales pour mémoriser les étiquettes de langage. Afin de permettre son ajout dans une version future de la présente norme, les interpréteurs UTF-8 côté client DOIVENT être capables d'ignorer en silence les caractères multi-octets UTF-8 illégaux, et de traiter les caractères UTF-8 illégaux d'un seul octet comme des marqueurs de fin de chaîne. Pour l'instant, les serveurs DOIVENT être capables d'accepter en silence les caractères UTF-8 illégaux, sauf dans les noms d'attributs et d'entrées. Les clients NE DOIVENT PAS envoyer de caractères UTF-8 illégaux au serveur sauf si une norme future changeait cette règle.

(2) Il est proposé d'ajouter des étiquettes de langage à Unicode. Pour prendre cela en charge, les serveurs DOIVENT être capables de mémoriser les caractères UTF-8 jusqu'à 20 bits de données.

(3) L'élément de métadonnées "language" est réservé pour utilisation future.

10. Considérations pour la sécurité

La commande AUTHENTICATE utilise SASL [RFC2222] pour fournir les services d'authentification de base, d'autorisation, d'intégrité et de confidentialité. Ceci est décrit au paragraphe 6.3.1.

Lorsque le mécanisme CRAM-MD5 est utilisé, les considérations pour la sécurité du mécanisme SASL de CRAM-MD5 [RFC2195] s'appliquent. Le mécanisme CRAM-MD5 est aussi susceptible d'attaques de dictionnaire passives. Cela signifie que si une session d'authentification est enregistrée par un observateur passif, cet observateur peut essayer les mots de passe courants sur le mécanisme CRAM-MD5 pour voir si le résultat correspond. Cette attaque est déjouée en utilisant des mots de passe difficiles à deviner. Les sites sont invités à éduquer les usagers et à confronter les candidats mots de passe à un dictionnaire avant de les mettre en service. Les mises en œuvre ACAP de CRAM-MD5 DEVRAIENT permetre des mots de passe d'au moins 64 caractères de long.

Les transactions du protocole ACAP sont susceptibles d'attaques d'observation passive ou par interposition *(man in the middle)* qui altèrent les données, à moins que les services facultatifs de chiffrement et d'intégrité de la commande AUTHENTICATE ne soient activées, ou qu'un mécanisme externé de sécurité ne soit utilisé pour la protection. Il peut être utile de permettre la configuration aussi bien des clients que des serveurs pour refuser de transférer des informations sensibles en l'absence de chiffrement fort.

Les listes de contrôle d'accès de ACAP fournissent des autorisations raffinées pour l'accès aux attributs. Un certain nombre de questions en rapport avec la sécurité sont décrites au paragraphe 3.5.

Les URL ACAP ont les mêmes considérations de sécurité que les URL IMAP [RFC2192].

Les clients ACAP sont invités à considérer les problèmes de sécurité impliqués par la situation d'un ordinateur de laboratoire. Précisément, une antémémoire client d'informations de configuration ACAP NE DOIT PAS permettre l'accès à des usagers non autorisés. Une façon de le garantir est qu'un client ACAP soit capable de purger complètement toutes les données de configuration non publiques en antémémoire lorsque un usager le quitte.

Comme les ordinateurs portables sont facilement volés et qu'une antémémoire de données de configuration peut contenir des informations sensibles, un client ACAP en mode déconnecté peut souhaiter chiffrer et protéger par un mot de passe les informations de configuration en antémémoire.

11. Remerciements

Tous nos remerciements aux personnes suivantes qui ont contribué à ACAP au fil des quatre dernières années : Wallace Colyer, Mark Crispin, Jack DeWinter, Rob Earhart, Ned Freed, Randy Gellens, Terry Gray, J. S. Greenfield, Steve Dorner, Steve Hole, Steve Hubert, Dave Roberts, Bart Schaefer, Matt Wall et tous les autres participants au groupe de travail ACAP de l'IETF.

12. Adresse des auteurs

Chris Newman
Innosoft International, Inc.
1050 Lakes Drive
West Covina, CA 91790
USA
mél : chris.newman@innosoft.com

John Gardiner Myers
Netscape Communications
501 East Middlefield Road
Mail Stop MV-029
Mountain View, CA 94043
mél : jgmyers@netscape.com

Appendice A. Références

[ISO-10646] ISO/IEC 10646-1:1993(E) "Information Technology-- Universal Multiple-octet Coded Character Set (UCS)." Voir aussi les amendements 1 à 7, plus les corrections rédactionnelles.

[ISO-C] ISO/IEC 9899:1990, "Programming languages -- C", International Organization for Standardization. C'est la même chose que la norme ANSI C X3.159-1989.

[RFC1738] T. Berners-Lee et autres, "Localisateurs uniformes de ressource (URL)", décembre 1994. *(P.S., Obsolète, voir les*

RFC4248 et 4266)

[RFC1766] H. Alvestrand, "Étiquettes pour l'identification des langues", mars 1995. *(Obsolète, voir* RFC3066, RFC3282) *(P.S.)*

[RFC1808] R. Fielding, "Localisateurs relatifs de ressource uniforme", juin 1995. (*Obsolète, voir* RFC3986)

[RFC2044] F. Yergeau, "UTF-8, un format de transformation 'Unicode et d'ISO 10646", octobre 1996. (*Obsolète, voir* RFC3629) *(D.S.)*

[RFC2060] M. Crispin, "Protocole d'accès au message Internet - version 4, rev1", décembre 1996. *(Remplace* RFC1730) (*Obsolète, voir* RFC3501) *(P.S.)*

[RFC2086] J. Myers, "Extension IMAP4 ACL", janvier 1997. (*Obsolète, voir* RFC4314) *(P.S.)*

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

[RFC2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin, P. Svanberg, "Rapport de l'atelier Jeux de caractères de l'IAB tenu du 29 février au 1er mars 1996", avril 1997. *(Information)*

[RFC2192] C. Newman, "Schéma d'URL IMAP", septembre 1997. (*Obsolète, voir* RFC5092) *(P.S.)*

[RFC2195] J. Klensin et autres, "Extension IMAP/POP AUTHorize pour mise au défi/réponse simple", septembre 1997. *(P.S.)*

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

[RFC2234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", novembre 1997. (*Obsolète, voir* RFC5234)

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

[RFC2277] H. Alvestrand, "Politique de l'IETF en matière de jeux de caractères et de langages", BCP 18, janvier 1998.

[UNICODE-2] The Unicode Consortium, "The Unicode Standard, Version 2.0", Addison-Wesley, 1996. ISBN 0-201-48345-9.

[US-ASCII] "USA Standard Code for Information Interchange," X3.4. American National Standards Institute: New York (1968).

Appendice B. Index des mots clés d'ACAP

Appendice C. Déclaration complète de droits de reproduction