Groupe de travail Réseau
Request for Comments : 3659
RFC mise à jour : 959
Catégorie : En cours de normalisation

P. Hethmon, Hethmon Software
mars 2007

Traduction Claude Brière de L'Isle

Extensions à FTP

Résumé
Le présent document spécifie de nouvelles commandes FTP pour obtenir des listes de répertoires distants dans un format défini , et pour permettre le redémarrage de transferts de données interrompus en mode STREAM. Il permet des jeux de caractères autres que l'US-ASCII, et il définit aussi une structure facultative de mémorisation de fichier virtuel.

Table des Matières

1. Introduction

Le présent document met à jour le protocole de transfert de fichier (FTP, *File Transfer Protocol*) [RFC959]. Quatre nouvelles commandes sont ajoutées : "SIZE", "MDTM", "MLST", et "MLSD". La commande existante "REST" est modifiée. Parmi celles-ci, les commandes "SIZE" et "MDTM" et les modifications à "REST" ont été de large utilisation pendant de nombreuses années. Les autres sont nouvelles.

Ces commandes permettent à un client de redémarrer un transfert interrompu dans des modes de transfert non pris en charge précédemment de façon documentée, et d'obtenir un répertoire dans un format prévisible lisible par la machine.

Une structure facultative pour la mémorisation des fichiers du serveur (NVFS) est aussi définie, permettant aux serveurs qui prennent en charge une telle structure de porter ces informations aux clients d'une façon standard, permettant donc aux clients plus de certitude pour construire et interpréter les noms de chemin.

2. Conventions du document

Le présent document utilise les conventions de document définies dans le BCP 14, [RFC2119]. Cela donne l'interprétation des mots impératifs en majuscule comme DOIT, DEVRAIT, etc.

Le présent document utilise aussi la notation définie dans le STD 9, [RFC959]. En particulier, les termes "réponse", "utilisateur", "NVFS" (système réseau de fichier virtuel, *Network Virtual File System*), "fichier", "nom de chemin", "commandes FTP", "DTP" (processus de transfert de données, *data transfer process*), "processus d'usager FTP", "usager-PI" (interpréteur de protocole d'usager), "usager-DTP", "processus de serveur FTP", "serveur-PI", "serveur-DTP", "mode", "type", "NVT" (terminal virtuel réseau , *Network Virtual Terminal*), "connexion de contrôle", "connexion de données", et "ASCII", sont tous utilisés ici selon ses définitions.

La syntaxe requise est définie en utilisant le formalisme Backus Naur augmenté (ABNF) défini dans la [RFC4234]. Certaines définitions générales d'ABNF qui sont requises tout au long du document seront définies plus loin dans cette section. En première lecture, il peut être sage de simplement se rappeler que ces définitions existent, et de sauter à la section suivante.

2.1 Jetons de base

Le présent document importe le cœur des définitions ABNF données à l'Appendice A de la [RFC4234]. Ces définitions se trouvent pour les éléments de base de l'ABNF tels que ALPHA, CHIFFRE, SP, etc. Les termes suivants sont ajoutés pour être utilisés dans le présent document.

```
TCHAR = VCHAR / SP / HTAB                ; caractères visible plus les espaces
RCHAR = ALPHA / CHIFFRE / "," / "." / ":" / "!" / "@" / "#" / "$" / "%" / "^" / "&" / "(" / ")" / "-" / "_" / "+" / "?" / "/" / "\"
        / "'" / DQUOTE                   ; <"> -- c'est le caractère guillemets (%x22)
SCHAR = RCHAR / "=" ;
```

Les types VCHAR (de la [RFC4234]) RCHAR, SCHAR, et TCHAR donnent les types de caractères de base de divers sous-ensembles du jeu de caractères ASCII à utiliser dans diverses commandes et réponses.

```
jeton = 1*RCHAR
```

Un "jeton" *(token)* est une chaîne dont la signification précise dépend du contexte dans lequel elle est utilisée. Dans certains cas, ce sera une valeur tirée d'un ensemble de valeurs possibles tenu ailleurs. Dans d'autres, ce peut être une chaîne inventée par un participant à une conversation FTP à partir de la source qu'il aura estimée pertinente.

Noter qu'en ABNF, les littéraux de chaîne sont insensibles à la casse. Cette convention est conservée dans le présent document, et elle implique que les commandes FTP ajoutées par la présente spécification ont des noms qui peuvent être représentés dans toute casse. C'est à dire que "MDTM" est la même chose que "mdtm", "Mdtm" et "MdTm" etc.. Cependant, noter que ALPHA, en particulier, est sensible à la casse. Cela implique que un "jeton" est une valeur sensible à la casse. Cette implication est correcte, sauf mention explicite contraire dans le présent document, ou dans quelque autre spécification qui définit les valeurs que spécifie le présent document pour être utilisées dans un contexte particulier.

2.2 Noms de chemin

Diverses commandes FTP prennent des noms de chemin comme arguments, ou retournent des noms de chemin en réponse. Lorsque la commande MLST est prise en charge, comme indiqué dans la réponse à la commande FEAT [RFC2389], les noms de chemin sont à transférer sous l'un des deux formats suivants.

```
nom_de_chemin   = nom_utf-8 / brut
nom_utf-8       = <chaîne Unicode codée en UTF-8>
brut            = <toute chaîne qui n'est pas un codage UTF-8 valide>
```

Le format à utiliser est au choix de l'usager-PI ou du serveur-PI qui envoie le nom de chemin. Les codages UTF-8 [RFC3629] contiennent assez de structure interne pour qu'il soit toujours, en pratique, possible de déterminer si un codage UTF-8 ou brut a été utilisé, dans les cas où cela importe. Alors qu'il est utile à l'usager-PI d'être capable de correctement afficher un nom de chemin reçu du serveur-PI par l'usager, il est bien plus important pour l'usager-PI d'être capable de conserver et de retransmettre le nom de chemin identique quand nécessaire. Il est conseillé aux mises en œuvre de ne pas convertir un nom de chemin UTF-8 dans un jeu de caractères local qui ne serait pas capable de représenter la totalité du répertoire de caractères Unicode, et de tenter d'inverser ultérieurement la traduction du jeu de caractères. Noter que l'ASCII est un sous ensemble de l'UTF-8. Voir aussi [ANSI-X3.4].

Sauf spécification contraire, le nom de chemin se termine par le CRLF qui termine la commande FTP, ou par le CRLF qui termine une réponse. Toutes espaces en queue précédant ce CRLF font partie du nom. Exactement une espace va précéder le nom de chemin et va servir de séparateur d'avec le précédant élément syntaxique. Toutes les espaces supplémentaires font partie du nom de chemin. Voir la [RFC2640] pour une explication plus complète des questions de codage de caractère. Toutes les mises en œuvre qui prennent en charge MLST DOIVENT prendre en charge la [RFC2640].

Note : pour les noms de chemin transférés sur une connexion de données, il n'existe pas de moyen pour représenter un nom de chemin qui contient les caractères CR et LF à la suite, et de distinguer cela de l'indication d'une fin de ligne. Donc, les noms de chemin qui contiennent la paire de caractères CRLF ne peuvent pas être transmis sur une connexion de données. Les connexions de données contiennent seulement des noms de fichiers transmis du serveur FTP à l'usager FTP en résultat d'une des commandes de liste de répertoire. Les fichiers qui ont des noms contenant la séquence CRLF doivent soit avoir cette séquence convertie en une autre forme, telle qu'elle puisse être reconnue et être correctement reconvertie en CRLF, soit être omis de la liste.

Les mises en œuvre devraient aussi veiller à ce que la connexion de contrôle FTP utilise les conventions Telnet NVT de la [RFC854], et que le caractère Telnet IAC, si il fait partie d'un nom de chemin envoyé sur la connexion de contrôle, DOIT être correctement échappé, comme défini par le protocole Telnet.

NVT distingue aussi entre CR, LF, et le CRLF de fin de ligne, et permettrait ainsi que des noms de chemin contenant la paire de caractères CR et LF soient correctement transmis. Cependant, comme une telle séquence ne peut pas être transmise sur une connexion de données (au titre du résultat d'une commande LIST, NLST, ou MLSD) de tels noms de chemin sont à éviter.

Les mises en œuvre devraient aussi veiller à ce que, bien que les conventions Telnet NVT soient utilisées sur les connexions de contrôle, la négociation d'option Telnet NE DOIT PAS être tentée. Voir le paragraphe 4.1.2.12 de la [RFC1123].

2.2.1 Syntaxe de nom de chemin

Sauf lorsque TVFS est pris en charge (voir la section 6) la présente spécification n'impose aucune syntaxe aux noms de chemins. Elle ne restreint pas non plus le jeu de caractères dans lequel est créé le nom de chemin. Cela n'implique pas que

le NVFS est obligé de donner un sens à tous les noms de chemins possibles. Les serveurs-PI peuvent restreindre la syntaxe des noms de chemin valides dans leur NVFS de toute façon appropriée à leur mise en œuvre ou à leur système de fichiers sous-jacent. De même, un serveur-PI peut analyser le nom de chemin et allouer une signification aux composants détectés.

2.2.2 Caractères génériques

Pour les commandes définies dans la présente spécification, tous les noms de chemins sont à traiter littéralement. C'est-à-dire, pour un nom de chemin donné comme un paramètre pour une commande, le fichier dont le nom est identique au nom de chemin donnée est impliqué. Aucun caractère du nom de chemin ne peut être traité comme spécial ou "magique", donc, aucune correspondance de schéma (autre que d'égalité exacte) entre le nom de chemin donné et les fichiers présents dans le NVFS du serveur-FTP n'est permise.

Les clients qui désirent une forme de fonctionnalité de correspondance de schéma doivent obtenir une liste du ou des répertoires pertinents, et mettre en œuvre leurs propres procédures de sélection de noms de chemin.

2.3. Date et heure

La syntaxe d'une valeur d'heure *(time-val)* est :

```
valeur_d'heure          = 14CHIFFRE [ "." 1*CHIFFRE ]
```

Les quatorze chiffres de tête, obligatoires, sont à interpréter, dans l'ordre de gauche à droite, comme quatre chiffres donnant l'année, dans une gamme de 1000 à 9999, deux chiffres pour le mois de l'année, dans une gamme de 01 à 12, deux chiffres donnant le jour du mois, dans une gamme de 01 à 31, deux chiffres donnant l'heure du jour, dans une gamme de 00 à 23, deux chiffres donnant la minute dans l'heure, dans une gamme de 00 à 59, et finalement, deux chiffres qui donnent la seconde dans la minute, dans une gamme de 00 à 60 (60 n'étant utilisé que pour un saut de seconde). Les années dans le dixième siècle, et avant, ne peuvent pas être exprimées. Ce défaut n'est pas considéré comme sérieux pour le protocole.

Les chiffres facultatifs, qui sont précédés d'un point, donnent une fraction décimale de seconde. Elle peut être donnée à toute précision appropriée aux circonstances, cependant les mises en œuvre NE DOIVENT PAS ajouter de précision aux valeurs d'heure lorsque cette précision n'existe pas dans la valeur sous-jacente à transmettre.

Symboliquement, une valeur d'heure peut être vue comme : AAAAMMJJHHMMSS.sss

Le "." et les chiffres qui le suivent ("sss") sont facultatifs. Cependant, le "." NE DOIT PAS apparaître si au moins un chiffre n'apparaît ensuite.

Les valeurs de temps sont toujours représentées en UTC (GMT), et dans le calendrier grégorien sans considération du calendrier qui peut avoir été utilisé à la date et heure indiquées à la localisation du serveur-PI.

Les différences techniques entre GMT, TAI, UTC, UT1, UT2, etc., ne sont pas prises en compte ici. Un processus de serveur FTP devrait toujours utiliser la même référence temporelle, de sorte que les heures qu'il retourne soient cohérentes. Les clients ne sont pas supposés être synchronisés avec le serveur, de sorte que les possibles différences horaires qui pourraient être rapportées par les différentes normes horaires ne sont pas considérées comme importantes.

2.4 Réponses de serveur

Le paragraphe 4.2 de la [RFC959] définit le format et la signification des réponses du serveur-PI aux commandes FTP de la part de l'usager-PI. Ces conventions de réponse sont utilisées ici sans changement.

```
réponse_d'erreur        = code_d'erreur SP *TCHAR CRLF
code_d'erreur           = ("4" / "5") 2DIGIT
```

Les mises en œuvre devraient noter que la syntaxe ABNF utilisée dans le présent document et d'autres documents en rapport avec FTP (mais non utilisés dans la [RFC959]) montrent parfois des réponses utilisant un format sur une ligne. Sauf mention explicite contraire, cela n'est pas destiné à impliquer que des réponses sur plusieurs lignes ne sont pas permises. Les mises en œuvre devraient supposer que, sauf mention contraire, toute réponse à toute commande FTP (y compris QUIT) peut utiliser le format multiligne décrit dans la [RFC959].

Tout au long du présent document, les réponses seront identifiées par le code à trois chiffres qui est leur premier élément. Donc, le terme "réponse 500" signifie une réponse du serveur-PI qui utilise le code à trois chiffres "500".

2.5 Interprétation des exemples

Dans les exemples de dialogues FTP présentés dans le présent document, les lignes qui commencent par "C> " ont été envoyées sur la connexion de contrôle de l'usager-PI au serveur-PI, les lignes qui commencent par "S> " ont été envoyées sur la connexion de contrôle du serveur-PI à l'usager-PI, et chaque séquence de lignes qui commence par "D> " a été envoyée du serveur-DTP à l'usager-DTP sur une connexion de données créée juste pour envoyer ces lignes et fermée immédiatement après. Aucun des exemples donnés ici ne montre de données transférées sur une connexion de données du client au serveur. Dans tous les cas, les préfixes montrés ci-dessus, y compris celui avec une espace, ont été ajoutés pour les besoins du présent document, et ne font pas partie des données échangées entre client et serveur.

3. Heure de modification de fichier (MDTM)

La commande FTP Heure de modification (MDTM, *MODIFICATION TIME*) peut être utilisée pour déterminer quand un fichier dans le serveur NVFS a été modifié pour la dernière fois. Cette commande a existé dans de nombreux serveurs FTP pendant de nombreuses années, comme adjonction à la commande REST pour le mode STREAM, et est donc largement disponible. Cependant, lorsque elle est prise en charge, le dispositif "modify" qui peut être fourni dans le résultat de la nouvelle commande MLST est recommandé comme solution supérieure de remplacement.

Lorsque on tente de redémarrer par une commande RETRieve *(récupérer)*, l'utilisateur FTP peut utiliser la commande MDTM ou le dispositif "modify" pour vérifier si l'heure de modification du fichier source est plus récente que l'heure de modification du fichier partiellement transféré. Si elle l'est, il est alors très vraisemblable que le fichier source a changé, et il ne serait pas sûr de redémarrer le transfert de fichier précédemment incomplet.

Comme les horloges de l'usager et du serveur FTP ne sont pas nécessairement synchronisées, les usagers FTP qui ont l'intention d'utiliser cette méthode devraient normalement obtenir l'heure de modification à partir du fichier du serveur avant le RETR initial, et la comparer à l'heure de modification avant un RESTart. Si elles diffèrent, les fichiers peuvent avoir changé, et RESTart ne serait pas conseillé. Lorsque ceci n'est pas possible, l'usager FTP devrait s'assurer de permettre un biais d'horloge lors de la comparaison des heures.

Lorsque on tente de redémarrer un STORe *(mémoriser)*, l'usager FTP peut utiliser la commande MDTM pour découvrir l'heure de modification du fichier partiellement transféré. Si il est plus vieux que l'heure de modification du fichier qui est sur le point d'être mémorisé, le plus vraisemblable est que le fichier source a changé, et il ne serait pas sûr de redémarrer le transfert.

Noter qu'utiliser MLST (décrit plus loin) lorsque disponible, peut fournir cette information et beaucoup plus, donnant ainsi une meilleure indication qu'un fichier a changé et que redémarrer un transfert ne donnerait pas des résultats valides.

Noter que ceci est applicable à toute tentative de RESTart, sans considération du mode du transfert de fichier.

3.1 Syntaxe

La syntaxe pour la commande MDTM est :

mdtm = "MdTm" SP nom_de_chemin CRLF

Comme avec toutes les commandes FTP, l'étiquette de commande "MDTM" est interprétée de manière insensible à la casse.

Le "nom_de_chemin" spécifie un objet dans le NVFS qui peut être l'objet d'une commande RETR. Les tentatives d'interrogation de l'heure de modification des fichiers qui existent mais qu'on ne peut pas récupérer peuvent générer une réponse d'erreur, ou peuvent résulter en une réponse positive qui porte une valeur_horaire non spécifiée, le choix étant fait par le serveur-PI.

Le serveur-PI va répondre à la commande MDTM par une réponse 213 qui donne l'heure de la dernière modification du fichier dont le nom de chemin a été fourni, ou une réponse 550 si le fichier n'existe pas, si l'heure de modification est indisponible, ou si quelque autre erreur est survenue.

réponse-mdtm = "213" SP valeur_horaire CRLF / réponse_d'erreur

Noter que lorsque la réponse 213 est produite, c'est-à-dire, lorsque il n'y a pas d'erreur, le format DOIT être exactement comme spécifié. Les réponses multilignes ne sont pas permises.

3.2 Réponses d'erreur

Lorsque la commande est analysée correctement mais que l'heure de modification n'est pas disponible, soit parce que le nom de chemin n'identifie aucune entité existante, soit parce que les informations ne sont pas disponibles pour l'entité désignée, une réponse 550 devrait alors être envoyée. Lorsque la commande ne peut pas être analysée correctement, une réponse 500 ou 501 devrait être envoyée, comme spécifié dans la [RFC959]. Diverses réponses 4yz sont aussi possibles dans les circonstances appropriées.

3.3 Réponse FEAT pour MDTM

Lorsque il répond à la commande FEAT [RFC2389], un processus de serveur FTP qui prend en charge la commande MDTM DOIT inclure une ligne contenant le seul mot "MDTM". Cela PEUT être envoyé en majuscules ou en minuscules ou dans un mélange des deux (ce n'est pas sensible à la casse) mais DEVRAIT n'être transmis qu'en majuscules. C'est à dire que la réponse DEVRAIT être :

C> Feat
S> 211- <n'importe quel texte descriptif>
S> ...
S> MDTM
S> ...
S> 211 Fin

Les chevrons (< >) indiquent un bouche-trou où d'autres caractéristiques peuvent être incluses, mais ne sont pas exigées. L'indentation d'une espace des lignes de caractéristiques est obligatoire selon la [RFC2389].

3.4 Exemples de MDTM

Si on suppose l'existence de trois fichiers, A, B et C, un répertoire D, deux fichiers avec des noms qui se terminent par la chaîne "ile6", et aucun autre fichier, alors la commande MDTM peut se comporter comme indiqué. Les lignes "C>" sont des commandes de l'usager-PI au serveur-PI, les lignes "S>" sont les réponses du serveur-PI.

C> MDTM A
S> 213 19980615100045.014
C> MDTM B
S> 213 19980615100045.014
C> MDTM C
S> 213 19980705132316
C> MDTM D
S> 550 D n'est pas récupérable
C> MDTM E
S> 550 Pas de fichier nommé "E"
C> mdtm fichier6
S> 213 19990929003355
C> MdTm 19990929043300 Fichier6
S> 213 19991005213102
C> MdTm 19990929043300 fichier6
S> 550 19990929043300 fichier6: Ce fichier ou répertoire n'existe pas.

À partir de cela, on peut conclure que A et B ont tous deux été modifiés pour la dernière fois à la même heure (à la milliseconde près) et que C a été modifié 20 jours et plusieurs heures après.

Les heures sont en GMT, de sorte que le fichier A a été modifié le 15 juin 1998, à approximativement 11 heures à Londres (l'heure d'été y était alors appliquée) ou peut-être 20 heures à Melbourne, Australie, ou à 6 heures New York. Toutes représentant, bien sûr, la même heure absolue. La localisation de la modification du fichier, et par conséquent, l'heure de l'horloge locale à cette localisation, n'est pas disponible.

Il n'y a pas de fichier nommé "E" dans le répertoire actuel, mais il y a un fichier nommé à la fois "fichier6" et "19990929043300 Fichier6". L'heure de modification de ces fichiers a été obtenue. Il n'y a pas de fichier nommé "19990929043300 fichier6".

4. Commande SIZE

La commande FTP Taille de fichier (SIZE) est utilisée pour obtenir du processus de serveur FTP la taille de transfert d'un fichier. C'est le nombre exact d'octets (de 8 bits) qui seraient transmis sur la connexion de données si ce fichier était transmis. Cette valeur va changer selon la STRUcture, le MODE, et le TYPE actuels de la connexion de données ou d'une connexion de données qui serait créée s'il en était créé une à cet instant là. Donc, le résultat de la commande SIZE dépend des paramètres STRU, MODE, et TYPE actuellement établis.

La commande SIZE retourne le nombre d'octets qui seraient transférés si le fichier était transféré en utilisant la structure, le mode, et le type de transfert actuels. Cette commande est normalement utilisée en conjonction avec la commande REDÉMARRER (REST) lorsque on mémorise (STOR) un fichier sur un serveur distant en mode STREAM, pour déterminer le point de redémarrage. Le serveur-PI pourrait avoir besoin de lire le fichier partiellement transféré, de faire toute conversion appropriée, et de compter le nombre d'octets qui seraient générés lors de l'envoi du fichier afin de répondre correctement à cette commande. Les estimations de la taille du transfert de fichier NE DOIVENT PAS être retournées ; seules des informations précises sont acceptables.

4.1 Syntaxe

La syntaxe de la commande SIZE est :

size = "taille" SP nom_de_chemin CRLF

Le serveur-PI va répondre à la commande SIZE par un 213 qui donne la taille du transfert du fichier dont le nom de chemin a été fourni, ou une réponse d'erreur si le fichier n'existe pas, si la taille est indisponible, ou si une autre erreur est survenue. La valeur retournée est dans un format convenable pour l'utilisation avec la commande RESTART (REST) pour le mode STREAM, pourvu que le mode et le type de transfert ne soient pas altérés.

réponse_de_taille = "213" SP 1*CHIFFRE CRLF / réponse_d'erreur

Noter que quand la réponse 213 est produite, c'est-à-dire, lorsque il n'y a pas d'erreur, le format DOIT être exactement comme spécifié. Les réponses multiligne ne sont pas permises.

4.2 Réponses d'erreur

Lorsque la commande est correctement analysée mais que la taille n'est pas disponible, peut-être par ce que le nom de chemin n'identifie aucune entité existante ou parce que l'entité désignée ne peut pas être transférée dans les MODE et TYPE actuels (ou pas du tout) une réponse 550 devrait alors être envoyée. Si la commande ne peut pas être analysée correctement, une réponse 500 ou 501 devrait être envoyée, comme spécifié dans la [RFC959]. La présence de la réponse d'erreur 550 à une commande SIZE NE DOIT PAS être prise par le client comme une indication que le fichier ne peut pas être transféré dans les MODE et TYPE actuels. Un serveur peut générer cette erreur pour d'autres raisons -- par exemple, si la redondance de traitement est considérée comme trop importante. Diverses réponses 4yz sont aussi possibles dans les circonstances appropriées.

4.3 Réponse FEAT pour SIZE

Lors d'une réponse à la commande FEAT de la [RFC2389], un processus de serveur FTP qui prend en charge la commande SIZE DOIT inclure une ligne contenant le seul mot "SIZE". Ce mot est insensible à la casse, et PEUT être envoyé dans tout mélange de majuscules ou minuscules ; cependant, il DEVRAIT être envoyé en majuscules. C'est-à-dire que la réponse DEVRAIT être :

C> FEAT
S> 211- <tout texte descriptif>
S> ...
S> SIZE
S> ...
S> 211 FIN

Les chevrons (< >) indiquent un bouche-trou où d'autres caractéristiques peuvent être incluses, mais ne sont pas exigées. L'indentation d'une espace des lignes de caractéristiques est obligatoire selon la [RFC2389].

4.4 Exemples de Size

Considérons un fichier texte "Exemple" mémorisé sur un serveur Unix(TM) où chaque extrémité de ligne est représentée par un seul octet. Supposons que le fichier contienne 112 lignes, et un total de 1830 octets. La commande SIZE produirait alors :

```
C> TYPE I
S> 200 Type réglé à I.
C> size Exemple
S> 213 1830

C> TYPE A
S> 200 Type réglé à A.
C> Size Exemple
S> 213 1942
```

Remarquer qu'avec TYPE=A la commande SIZE rapporte 112 octets supplémentaires. Ce sont les octets supplémentaires qui doivent être insérés, un à la fin de chaque ligne, pour fournir la sémantique correcte de fin de ligne pour un transfert utilisant TYPE=A. D'autres systèmes pourraient être nécessaires pour faire d'autres changements au format de transfert de fichiers lors d'une conversion de TYPE et de MODE. La commande SIZE prend tout cela en compte.

Comme le calcul de la taille d'un fichier avec ce degré de précision peut demander des efforts considérables de la part du serveur-PI, les utilisateurs-PI ne devraient pas utiliser cette commande sauf si cette précision est essentielle (comme lorsque on est sur le point de redémarrer un transfert interrompu). Pour d'autres usages, le fait "Size" de la commande MLST (voir au paragraphe 7.5.7) devrait être demandé.

5. Redémarrage de transfert interrompu (REST)

Pour éviter d'avoir à envoyer à nouveau le fichier entier si il est seulement partiellement transféré, les deux côtés ont besoin d'un moyen pour s'accorder sur l'endroit du flux de données où redémarrer le transfert.

La spécification FTP de la [RFC959] comporte trois modes de transfert de données, STREAM, Bloc, et Compressé. Dans les modes Bloc et Compressé, le flux de données qui est transféré sur la connexion de données est formaté, ce qui permet l'incorporation de marqueurs de redémarrage dans le flux. Le DTP d'envoi peut inclure un marqueur de redémarrage avec toutes les informations dont il a besoin pour être capable de redémarrer un transfert de données à ce point. Le DTP receveur peut conserver une liste de ces marqueurs de redémarrage, et les corréler avec ce qui est sauvegardé du fichier. Pour redémarrer le transfert de fichier, le receveur renvoie juste le dernier marqueur de redémarrage, et les deux côtés savent comment reprendre le transfert de données. Noter qu'il y a quelques fautes dans la description du mécanisme dans le STD 9, [RFC959]. Voir les corrections au paragraphe 4.1.3.4 de la [RFC1123].

5.1 Redémarrage en mode STREAM

En mode STREAM, la connexion de données contient juste un flux d'octets de données non formatées. Des marqueurs explicites de redémarrage ne peuvent donc pas être insérés dans le flux des données, car on ne pourrait pas les distinguer des données. Pour cette raison, la spécification FTP [RFC959] ne fournissait pas de capacité d'effectuer de redémarrage en mode flux. Cependant, il n'y a pas réellement de besoin d'avoir des marqueurs explicites de redémarrage dans ce cas, car des marqueurs de redémarrage peuvent être impliqués par le décalage d'octets dans le flux de données.

Parce qu'en mode flux, le flux de données définit le fichier, un flux de données différent représenterait un fichier différent. Donc, un décalage va toujours représenter la même position au sein d'un fichier. D'un autre côté, dans d'autres modes que STREAM, le même fichier peut être transféré en utilisant des séquences d'octets assez différentes et être quand même reconstruit en un fichier identique. Donc, un décalage dans le flux de données dans les modes de transfert autres que STREAM ne donnerait pas un point de redémarrage non ambigu.

Si le type de représentation des donnée est IMAGE et si la structure est File, pour de nombreux systèmes, le fichier sera mémorisé exactement sous le même format que si il est envoyé à travers la connexion de données. Il est alors habituellement très facile au receveur de déterminer quelle quantité de données a été reçue précédemment, et de notifier à l'envoyeur le décalage où le transfert devrait être redémarré. Dans d'autres types et structures de représentation, il faudra plus d'efforts, mais il reste toujours possible de déterminer le décalage avec des efforts définissables, mais peut-être non négligeables. Dans le pire des cas, un processus FTP peut avoir besoin d'ouvrir une connexion de données avec lui-même, de régler le type et la structure de transfert appropriés, et de transmettre effectivement le fichier en comptant les octets transmis.

Si le processus d'utilisateur-FTP est destiné à redémarrer une restitution, il va directement calculer le marqueur de redémarrage et envoyer ces informations dans la commande RESTart. Cependant, si le processus d'utilisateur FTP est destiné à redémarrer l'envoi du fichier, il a besoin d'être capable de déterminer quelle quantité de données ont été envoyées précédemment, et combien ont été correctement reçues et sauvegardées. Une nouvelle commande FTP est nécessaire pour obtenir des informations. C'est l'objet de la commande SIZE, comme documentée dans la section 4.

5.2 Récupération d'erreur et redémarrage

Les transferts en mode STREAM avec la structure FILE peuvent être redémarrés même si aucun marqueur de redémarrage n'a été transféré en plus des données elles-mêmes. Cela se fait en utilisant la commande SIZE, si nécessaire, en combinaison avec la commande REST, et une des commandes standard de transfert de fichier.

Lorsque on utilise le type ASCII ou IMAGE, la commande SIZE va retourner le nombre d'octets qui auraient été réellement transférés si le fichier devait être envoyé entre les deux systèmes, c'est-à-dire, avec le type IMAGE, la taille serait normalement le nombre d'octets du fichier. Avec le type ASCII, la taille serait le nombre d'octets dans le fichier incluant toutes les modifications nécessaires pour satisfaire à la convention CR-LF de fin de ligne du TYPE ASCII.

5.3 Syntaxe

La syntaxe pour la commande REST lorsque le mode de transfert en cours est STREAM est :

rest = "Rest" SP 1*CHIFFRE CRLF

La valeur numérique donne le nombre d'octets du transfert qui suit immédiatement et qui ne sont en fait pas envoyés, causant effectivement le redémarrage de la transmission à un point ultérieur. Une valeur de zéro désactive effectivement le redémarrage, causant la transmission du fichier entier. Le serveur-PI va répondre à la commande REST par un 350, indiquant que le paramètre REST a été sauvegardé, et qu'une autre commande, qui devrait être RETR ou STOR, devrait ensuite suivre pour achever le redémarrage.

réponse_de_redémarrage = "350" SP *TCHAR CRLF / réponse_d'erreur

Les processus de serveur FTP peuvent permettre des commandes de transfert autres que RETR et STOR, telles que APPE et STOU, pour achever un redémarrage ; cependant, ceci n'est pas recommandé. STOU (STOre Unique) est indéfini dans cet usage, car la mémorisation du reste d'un fichier dans un unique nom de fichier va rarement être utile. Si APPE (append, *ajouter*) est permis, il DOIT agir de façon identique à STOR lorsque un marqueur de redémarrage a été établi. C'est-à-dire que dans les deux cas, les octets provenant de la connexion de données sont placés dans le fichier à l'endroit indiqué par la valeur du marqueur de redémarrage.

La commande REST est destinée à achever un transfert qui avait échoué. L'utilisation avec RETR est comparativement bien définie dans tous les cas, car le client porte la responsabilité de la fusion des données récupérées avec le fichier partiellement récupéré. Il peut choisir d'utiliser les données obtenues plutôt que d'achever un transfert antérieur, ou de récupérer les données qui avaient été récupérées antérieurement. Avec STOR, cependant, le serveur doit plutôt insérer les données dans le fichier désigné. Les résultats sont indéfinis si un client utilise REST pour faire autre chose que redémarrer pour achever le transfert d'un fichier qui avait échoué précédemment à se transférer complètement. En particulier, si le marqueur de redémarrage établi avec une commande REST n'est pas à la fin des données actuellement mémorisées chez le serveur, comme le rapporte le serveur, ou si des données insuffisantes sont fournies dans un STOR qui suit un REST pour étendre le fichier de destination à au moins sa taille précédente, les effets sont alors indéfinis.

La commande REST doit être la dernière commande produite avant la commande de transfert de données qui va causer un redémarrage, plutôt qu'un transfert complet de fichier. L'effet de la production d'une commande REST à tout autre moment est indéfini. Le serveur-PI peut réagir à une commande REST mal positionnée en produisant une réponse d'erreur à la commande suivante, qui ne serait pas une commande de transfert de données redémarrable, ou il peut sauvegarder la valeur de redémarrage et l'appliquer à la prochaine commande de transfert de données, ou il peut ignorer en silence la tentative inapproprié de redémarrage. À cause de cela, un usager-PI qui a produit une commande REST, mais qui n'a pas réussi à transmettre la commande de transfert de données suivante pour une raison quelconque, devrait envoyer une autre commande REST avant la prochaine commande de transfert de données. Si ce transfert n'est pas à redémarrer, c'est alors "REST 0" qui devrait être produit.

Une réponse d'erreur va suivre une commande REST seulement lorsque le serveur ne met pas en œuvre la commande, ou lorsque la valeur du marqueur de redémarrage est syntaxiquement invalide pour le mode de transfert actuel (par exemple, en mode STREAM, quelque chose d'autre qu'un ou plusieurs chiffres apparaît dans le paramètre de la commande REST). Toutes les autres erreurs, y compris des problèmes tels qu'un marqueur de redémarrage hors gamme, devraient être

rapportés lors de la production de la prochaine commande de transfert. De telles erreurs vont être cause du rejet d'une demande de transfert avec une erreur indiquant la tentative invalide de redémarrage.

5.4 Réponse FEAT pour REST

Lorsque un processus de serveur FTP prend en charge RESTart en mode STREAM, comme spécifié ici, il DOIT inclure, dans la réponse à la commande FEAT [RFC2389], une ligne contenant exactement la chaîne "REST STREAM". Cette chaîne n'est pas sensible à la casse, mais elle DEVRAIT être transmise en majuscules. Lorsque REST n'est pas pris en charge du tout ou n'est pris en charge qu'en mode bloc ou compressé, la ligne REST NE DOIT PAS être incluse dans la réponse FEAT. Lorsque elle est exigée, la réponse DEVRAIT être :

C> feat
S> 211- <tout texte descriptif>
S> ...
S> REST STREAM
S> ...
S> 211 end

Les chevrons (< >) indiquent un bouche-trou où d'autres caractéristiques peuvent être incluses, mais ne sont pas exigées. L'indentation d'une espace des lignes de caractéristiques est obligatoire selon la [RFC2389].

5.5 Exemple de REST

On suppose que le transfert du fichier assez grand a précédemment été interrompu après que 802 816 octets ont été reçus, que le transfert précédent était avec TYPE=I, et qu'il a été vérifié que le fichier sur le serveur n'a pas été changé depuis.

C> TYPE I
S> 200 Type réglé à I.
C> PORT 127,0,0,1,15,107
S> 200 commande PORT réussie.
C> REST 802816
S> 350 Redémarrer à 802816. Envoi de STORE ou RETRIEVE
C> RETR cap60.pl198.tar
S> 150 Ouverture d'une connexion de données en mode BINAIRE
[...]
S> 226 Transfert terminé.

6. Mémorisation triviale de fichier virtuel (TVFS)

Traditionnellement, FTP n'a placé presque aucune contrainte sur la mémorisation de fichier (NVFS) fournie par un serveur. La présente spécification ne change pas cela. Cependant, il est devenu courant pour les serveurs de tenter de fournir au moins des conventions de dénomination de système de fichiers modelées librement sur celles du système de fichiers UNIX(TM). C'est un système de fichier structuré en arborescence, constitué de répertoires, dont chacun peut contenir d'autres répertoires, ou d'autres sortes de fichiers, ou les deux. Chaque fichier et répertoire a un nom par rapport au répertoire qui le contient, sauf pour le répertoire à la racine de l'arborescence, qui n'est contenu dans aucun autre répertoire, et donc n'a pas de nom à lui.

Ce qui a été décrit jusqu'à présent est parfaitement cohérent avec les mécanismes d'accès et le NVFS standard FTP. La commande "CWD" est utilisée pour se déplacer d'un répertoire à un répertoire incorporé. "CDUP" peut être fourni pour retourner au répertoire parent, et les diverses commandes de manipulation de fichier (les commandes "RETR", "STOR", et renommer, etc.) sont utilisées pour manipuler les fichiers au sein du répertoire courant.

Cependant, il est souvent utile d'être capable de faire référence à des fichiers autrement qu'en changeant de répertoire, en particulier parce que FTP ne donne aucun mécanisme garanti pour retourner à un répertoire précédent. Le protocole de mémorisation triviale de fichier virtuel (TVFS, *Trivial Virtual File Store*) s'il est mis en œuvre, fournit ce mécanisme.

6.1 Noms de fichier TVFS

Si un serveur met en œuvre le TVFS, aucun nom de fichier élémentaire ne doit contenir le caractère "/". Si la mémorisation naturelle sous-jacente de fichier permet aux fichiers, ou aux répertoires, de contenir le caractère "/" dans les noms, un serveur-PI qui met en œuvre TVFS doit coder ce caractère d'une certaine manière chaque fois que les noms de fichier ou de

répertoire sont retournés à l'usager-PI, et inverser ce codage chaque fois que ces noms sont acceptés par l'usager-PI.

La méthode de codage à utiliser n'est pas spécifiée ici. Si un autre caractère est illégal dans les noms de fichiers et répertoires de la mémorisation de fichiers sous-jacente, une simple translittération peut être suffisante. Si il n'y a pas de substitut convenable à ce caractère, un schéma de codage plus complexe, éventuellement en utilisant un caractère d'échappement, sera vraisemblablement nécessaire.

Avec la seule exception du répertoire racine sans nom, un nom de fichier TVFS ne peut pas être vide. C'est-à-dire que tous les autres noms de fichier doivent contenir au moins un caractère.

À la seule exception du caractère "/", tout caractère ISO 10646 [10646-1] peut être utilisé dans un nom de fichier TVFS. Lors de la transmission, les caractères du nom de fichier sont codés en utilisant le codage UTF-8 [RFC3629]. Noter que la séquence de deux caractères CR LF survenant dans un nom de fichier rendra ce nom impossible à transmettre sur une connexion de données. Par conséquent, cela devrait être évité, ou si il est impossible de faire autrement, il DOIT être codé d'une façon réversible.

6.2 Noms de chemin TVFS

Un "nom de chemin" TVFS combine le nom de fichier ou de répertoire d'un fichier ou répertoire cible, avec les noms de répertoire de zéro, un ou plusieurs répertoires englobants, afin de permettre au fichier ou répertoire cible d'être référencé autrement que quand le "répertoire de travail courant" du serveur est le répertoire qui contient directement le fichier ou répertoire cible.

Par définition, tout nom de fichier ou répertoire TVFS est aussi un nom de chemin TVFS. Un tel nom de chemin est valide pour référencer le fichier à partir du répertoire qui contient le nom, c'est-à-dire, lorsque ce répertoire est le répertoire de travail actuel du serveur FTP.

D'autres noms de chemin TVFS sont construits en ajoutant en préfixe à un nom de chemin un nom d'un répertoire sur lequel le chemin est valide, et en séparant les deux par le caractère "/". Un tel nom de chemin est valide pour référencer le fichier ou répertoire à partir du répertoire qui contient le nom de répertoire qui vient d'être ajouté.

Lorsque un nom de chemin a été étendu au point où le répertoire ajouté est le répertoire racine sans nom, le nom de chemin commencera par le caractère "/". Un tel chemin est appelé nom de chemin pleinement qualifié. Les chemins pleinement qualifiés ne peuvent évidemment pas être plus étendus, car, par définition, aucun répertoire ne contient le répertoire racine. N'ayant pas de nom, il ne peut être représenté dans aucun autre répertoire. Un nom de chemin pleinement qualifié est valide pour référencer le fichier ou répertoire désigné à partir de n'importe quelle localisation (c'est-à-dire, sans considération de ce que peut être le répertoire de travail actuel) dans la mémorisation de fichier virtuelle.

Tout nom de chemin qui n'est pas un nom de chemin pleinement qualifié peut être référencé comme "nom de chemin relatif" et ne fera correctement référence au fichier prévu que lorsque le répertoire de travail actuel du serveur FTP est un répertoire à partir duquel le nom de chemin relatif est valide.

Le cas particulier du nom de chemin "/" est défini comme étant un nom de chemin pleinement qualifié qui se réfère au répertoire racine. C'est-à-dire que le répertoire racine n'a pas de nom de répertoire (ou de fichier) mais qu'il a un nom de chemin. Ce nom de chemin spécial ne peut être utilisé comme il est que comme référence au répertoire racine. Il ne peut pas être combiné avec d'autres noms de chemin en utilisant les règles ci-dessus, car cela conduirait à un nom de chemin qui contiendrait deux caractères "/" consécutifs, ce qui est une séquence indéfinie.

6.2.1 Notes

Il n'est pas exigé, ni attendu, qu'il y ait seulement un nom de chemin pleinement qualifié qui référence un certain fichier ou répertoire particulier.

+ À titre indicatif, bien que la mémorisation de fichier TVFS soit à la base une structure arborescente, il n'est pas exigé qu'un fichier ou répertoire ait seulement un répertoire parent.
+ Comme on l'a défini, aucun nom de chemin TVFS ne va jamais contenir deux caractères "/" consécutifs. Un tel nom n'est cependant pas illégal, et peut être défini par le serveur pour tout objet qui lui convient. Les clients qui mettent en œuvre la présente spécification ne devraient pas supposer une sémantique pour de tels noms.
+ De même, en dehors des cas particuliers qui se réfèrent au répertoire racine, aucun nom de chemin TVFS construit comme défini ici ne va jamais se terminer par le caractère "/". De tels noms ne sont pas non plus illégaux, mais sont indéfinis.
+ Alors que tout caractère ISO 10646 légal est permis dans un nom de fichier ou répertoire TVFS, sauf "/", les mises en œuvre de serveur FTP ne sont pas obligées de prendre en charge tous les caractères ISO 10646 possibles. Le sous

ensemble pris en charge est entièrement à la discrétion du serveur. La casse (lorsque elle existe) des caractères qui constituent le nom du fichier, du répertoire, et du chemin peut être significative. Sauf détermination contraire par des moyens non spécifiés ici, les clients devraient supposer que de tels noms sont entièrement composés de caractères dont la casse est significative. Les serveurs ont toute liberté pour traiter la casse (ou tout autre attribut) d'un nom comme non pertinente, et donc de faire correspondre deux noms qui apparaissent comme étant distincts sous le même fichier sous-jacent.

+ Il n'y a pas de nom "magique" défini, comme ".", ".." ou "C:". Les serveurs peuvent mettre en œuvre de tels noms, avec la sémantique de leur choix, mais ils ne sont pas obligés de le faire.
+ TVFS n'impose pas de sémantique ni de propriété particulière aux fichiers, ne garantit aucun schéma de contrôle d'accès, ou aucune des autres propriétés communes d'une mémorisation de fichier. Seul le schéma de dénomination est défini.

6.3 Réponse FEAT pour TVFS

En réponse à la commande FEAT [RFC2389] un serveur qui souhaite indiquer qu'il prend en charge le TVFS comme défini ici va inclure une ligne qui commence par les quatre caractères "TVFS" (sous n'importe quelle casse, ou mélange de casses, les majuscules ne sont pas obligées). Les serveurs DEVRAIENT envoyer en majuscules.

Une telle réponse à la commande FEAT NE DOIT PAS être retournée sauf si le serveur met en œuvre TVFS comme défini ici.

Des spécifications ultérieures pourront ajouter à la définition de TVFS. De tels ajouts devraient être notifiés au moyen d'un texte supplémentaire ajouté à la ligne de caractéristiques TVFS. De telles spécifications, s'il en est, définiront le texte supplémentaire.

Jusqu'à ce qu'une telle spécification soit définie, les serveurs ne devraient rien inclure après le "TVFS" dans la ligne de caractéristiques TVFS. Cependant, les clients devraient être prêts à traiter un texte arbitraire à la suite des quatre caractères définis, et à simplement l'ignorer si il n'est pas reconnu.

Une réponse normale à la commande FEAT produite par un serveur qui met en œuvre seulement la présente spécification serait :

```
C> feat
S> 211- <tout texte descriptif>
S>  ...
S>  TVFS
S>  ...
S> 211 fin
```

Les chevrons (< >) indiquent un bouche-trou où d'autres caractéristiques peuvent être incluses, mais ne sont pas exigées. L'indentation d'une espace des lignes de caractéristiques est obligatoire selon la [RFC2389] et n'est pas compté comme un des quatre premiers caractères pour les besoins de cette liste de caractéristiques.

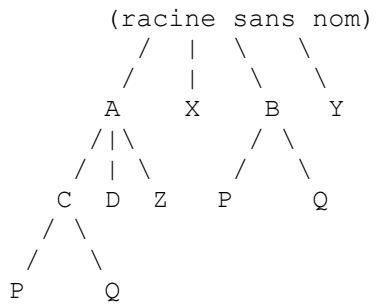
La caractéristique TVFS n'ajoute aucune nouvelle commande au répertoire des commandes FTP.

6.4 OPTS pour TVFS

Il n'y a pas d'option dans cette spécification TVFS, et donc, aucune commande OPTS n'est définie.

6.5 Exemples de TVFS

On suppose qu'une mémorisation de fichiers TVFS se compose d'un répertoire racine, qui contient deux répertoires (A et B) et deux fichiers non répertoires (X et Y). Le répertoire A contient deux répertoires (C et D) et un autre fichier (Z). Le répertoire B contient juste deux fichiers non répertoires (P et Q) et le répertoire C contient aussi deux fichiers non répertoires (aussi nommés P et Q, par hasard). Le répertoire D est vide, c'est-à-dire qu'il ne contient pas de fichier ou répertoire. Cette structure peut être décrite graphiquement comme ceci...



Dans cette structure, il existe les noms de chemin pleinement qualifiés suivants :

```
/
/A
/B
/X
/Y
/A/C
/A/D
/A/Z
/A/C/P
/A/C/Q
/B/P
/B/Q
```

Il est clair qu'aucun des chemins / /A /B ou /A/D ne se réfère au même répertoire, car le contenu de chacun est différent. Pas plus qu'aucun de / /A /A/C ou /A/D. Cependant /A/C et /B pourraient être le même répertoire, mais les informations sont insuffisantes pour le dire. Tous les autres noms de chemins (/X /Y /A/Z /A/C/P /A/C/Q /B/P et /B/Q) peuvent se référer aux mêmes fichiers sous-jacents, dans presque toutes les combinaisons.

Si le répertoire de travail actuel du serveur FTP est /A, alors les noms de chemins suivants, en plus de tous les noms de chemin pleinement qualifiés, sont valides :

```
C
D
Z
C/P
C/Q
```

Ils se réfèrent tous aux mêmes fichiers ou répertoires que le chemin pleinement qualifié correspondant avec "/A/" ajouté devant.

Que ces noms de chemin existent n'implique pas que le serveur TVFS va nécessairement accorder des droits d'accès aux chemins désignés, ou que l'accès au même fichier via des noms de chemin différents va nécessairement accorder des droits égaux.

Aucun des chemins relatifs suivant n'est valide lorsque le répertoire en cours est /A :

```
A
B
X
Y
B/P
B/Q
P
Q
```

Chacun d'eux pourrait être rendu valide en changeant le répertoire de travail actuel du serveur FTP en le répertoire approprié. Noter que les chemins "P" et "Q" pourraient se référer à des fichiers différents selon le répertoire qui est choisi pour faire qu'ils deviennent des chemins TVFS relatifs valides.

7. Listes pour traitement machine (MLST et MLSD)

Les commandes MLST et MLSD sont destinées à normaliser les informations de fichier et de répertoire retournées par le processus de serveur FTP. Ces commandes diffèrent de la commande LIST en ce que le format des réponses est strictement défini, bien qu'extensible.

Deux commandes sont définies, MLST et MLSD. MLST fournit des données sur exactement l'objet désigné sur sa ligne de commande, et aucune autre. L'autre, MLSD, fait la liste des contenus d'un répertoire si un répertoire est désigné, et autrement, une réponse 501 est retournée. Dans l'un ou l'autre cas, si aucun objet n'est désigné, le répertoire actuel est supposé. Cela va être cause que MLST envoie une réponse sur une ligne, qui décrit le répertoire actuel lui-même, et que MLSD fait la liste des contenus du répertoire actuel.

Dans ce qui suit, le terme MLSx sera utilisé lorsque MLST ou MLSD peut être inséré.

Les commandes MLST et MLSD étendent aussi le protocole FTP tel que présenté dans le STD 9, [RFC959] et le STD 3, [RFC1123] pour permettre la transmission de données de 8 bits sur la connexion de contrôle. Noter que cela n'est pas la spécification de jeux de caractères de 8 bits, mais cela spécifie que les mises en œuvre de FTP doivent spécifiquement permettre la transmission et la réception d'octets de 8 bits, dont tous les bits sont significatifs, sur la connexion de contrôle. C'est-à-dire que toutes les 256 valeurs d'octet possibles sont permises. La commande MLSx permet les deux formes UTF-8/Unicode et "brute" comme arguments, et en réponse, les deux commandes MLST et MLSD, et toutes les autres commandes FTP qui prennent des noms de chemin comme arguments.

7.1 Format des demandes MLSx

Les commandes MLST et MLSD permettent l'une et l'autre un seul argument facultatif. Cet argument peut être soit un nom de répertoire, soit, pour la seule MLST, un nom de fichier. À cette fin, un "nom de fichier" est le nom de toute entité dans le serveur NVFS qui n'est pas un répertoire. Lorsque TVFS est pris en charge, tout nom de chemin TVFS relatif valide dans le répertoire de travail courant, ou tout nom TVFS pleinement qualifié, peut être donné. Si un nom de répertoire est donné, la commande MLSD doit retourner une liste des contenus du répertoire désigné ; autrement, elle produit une réponse 501, et n'ouvre pas de connexion de données. Dans tous les cas pour MLST, un seul ensemble de lignes de faits (normalement une seule ligne de fait) contenant les informations sur le fichier ou répertoire désigné devra être retourné sur la connexion de contrôle, sans ouvrir une connexion de données.

Si aucun argument n'est donné, MLSD doit alors retourner une liste des contenus du répertoire de travail courant, et MLST doit retourner une liste qui donne des informations sur le répertoire de travail courant lui-même. À cette fin, les contenus d'un répertoire sont tous des noms de fichier ou répertoire (pas de noms de chemins) que le serveur-PI va permettre de référencer lorsque le répertoire de travail courant est le répertoire désigné, et que le serveur-PI désire révéler à l'usager-PI. Noter qu'omettre l'argument est la seule façon définie d'obtenir une liste du répertoire actuel, sauf si il se trouve qu'un nom de chemin qui représente le répertoire est connu. En particulier, il n'y a pas de nom abrégé défini pour le répertoire courant. Cela n'interdit pas qu'un serveur-PI particulier mette en œuvre un tel abrégé.

Aucun titre, en-tête, ou résumé, ligne, ou aucun autre élément de formatage, autres que ceux spécifiés ci-dessous, ne sont retournés dans le résultat d'une commande MLST ou MLSD.

Si le client-FTP envoie un argument invalide, le serveur-FTP DOIT répondre par un code d'erreur de 501.

La syntaxe de la commande MLSx est :

```
mlst        = "MLst" [ SP nom-de chemin ] CRLF
mlsd        = "MLsD" [ SP nom-de chemin ] CRLF
```

7.2 Format des réponses MLSx

Le format d'une réponse à une commande MLSx est le suivant :

```
réponse-mlst                = réponse-de-contrôle / réponse-d'erreur
réponse-mlsd                = ( réponse-initiale réponse-finale ) / réponse-d'erreur

réponse-de-contrôle         = "250-" [ message-de-réponse ] CRLF 1*( SP entrée CRLF )
                              "250" [ SP message-de-réponse ] CRLF

réponse-initiale            = "150" [ SP message-de-réponse ] CRLF
```

```
réponse-finale              = "226" SP message-de-réponse CRLF

message-de-réponse          = *TCHAR

réponse-de-données          = *( entrée CRLF )

entrée                      = [ faits ] SP nom-de-chemin
faits                       = 1*( fait ";" )
fait                = nom-de-fait "=" valeur
nom-de-fait         = "Taille" / "Modifie" / "Crée" / "Type" / "Unique" / "Perm" / "Langue" / "Type-de-support"
                      / "CharSet" / fait-dépendant-de-l'os / fait-local
fait-dépendant-de-l'os      = <Nom d'OS alloué par l'IANA> "." jeton
fait-local                  = "X." jeton
valeur                      = *SCHAR
```

À réception d'une commande MLSx, le serveur va vérifier le paramètre, et si il est invalide, retourner une réponse d'erreur. À cette fin, le paramètre devrait être considéré comme invalide si le client qui produit la commande n'a pas la permission d'effectuer l'opération demandée.

Si le paramètre est valide, pour une commande MLST, le serveur-PI va alors envoyer la première ligne (ligne directrice) de la réponse de contrôle, l'entrée pour le nom de chemin en question ou le répertoire actuel si aucun nom de chemin n'a été fourni, et la ligne de terminaison. Normalement, exactement une entrée devrait être retournée, plus d'entrées ne sont permises que lorsque c'est nécessaire pour représenter un fichier qui va avoir plusieurs faits "Type" retournés. Dans ce cas, le composant nom de chemin de chaque réponse DOIT être identique.

Noter que pour MLST, l'ensemble de faits est précédé par une espace. Cela est fait pour garantir que l'ensemble de faits ne peut pas être accidentellement interprété comme la ligne de terminaison de la réponse de contrôle, mais est exigé même lorsque cela ne serait pas possible. Exactement une espace existe entre l'ensemble de faits et le nom de chemin. Lorsque aucun fait n'est présent, il y aura exactement deux espaces en tête avant le nom de chemin. Aucune espace n'est permise dans les faits, toute autre espace dans la réponse est à traiter comme faisant partie du nom de chemin.

Si la commande était un MLSD, le serveur va ouvrir une connexion de données comme indiqué au paragraphe 3.2 du STD 9, [RFC959]. Si cela échoue, le serveur va retourner une réponse d'erreur. Si cela fonctionne, le serveur va retourner la réponse initiale, envoyer la réponse de données appropriée sur la nouvelle connexion de données, clore cette connexion, et ensuite envoyer la réponse finale sur la connexion de contrôle. La grammaire ci-dessus définit le format de la réponse de données, qui définit le format des données retournées sur la connexion de données établie.

La connexion de données ouverte pour une réponse MLSD devra être une connexion comme si les commandes "TYPE L 8", "MODE S", et "STRU F" avaient été données, quel que soit le type, le mode et la structure de transfert FTP qui ont été réellement établis, et sans causer de changement à ces réglages pour les futures commandes. C'est-à-dire que ce type de transfert devra être réglé pour la durée de la connexion de données établie seulement pour cette commande. Bien que le contenu des données envoyées puisse être vu comme une série de lignes, les mises en œuvre devraient noter qu'il n'y a pas de longueur maximum de ligne définie. Les mises en œuvre devraient être prêtes à traiter des lignes de longueur arbitraire.

La partie faits de la spécification va contenir une série de "faits de fichier" sur le fichier ou répertoire désigné sur la même ligne. Les informations normalement présentées vont inclure la taille du fichier, l'heure de la dernière modification, l'heure de création, un identifiant univoque, et un fanion de fichier/répertoire.

Le format complet d'une réponse réussie à la commande MLSD serait :

```
faits SP nom-de-chemin CRLF
faits SP nom-de-chemin CRLF
faits SP nom-de-chemin CRLF
...
```

Noter que le format est destiné au traitement machine, pas à la vue humaine, et à ce titre, le format est très rigide. Les mises en œuvre NE DOIVENT PAS faire varier le format, par exemple, en insérant des espaces supplémentaires pour la lisibilité, en remplaçant les espaces par des tabulations, en incluant des lignes d'en-tête ou de titre, ou en insérant des lignes blanches, ou de toute autre façon altérer ce format. Exactement une espace est toujours exigée après l'ensemble des faits (qui peut être vide). Plus d'espaces peuvent être présentes sur une ligne si, et seulement si, le nom de chemin présenté contient des espaces significatives. L'ensemble des faits ne doit pas contenir d'espaces en quelque endroit que ce soit. Les faits ne devraient être fournis dans chaque ligne de résultat que si ils fournissent des information pertinentes sur le fichier désigné sur la même ligne, et si ils sont dans l'ensemble demandé par l'usager-PI. Voir au paragraphe 7.9. Il n'est pas exigé que le même ensemble de faits soit fourni pour chaque fichier, ou que les faits présentés surviennent dans le même ordre

pour chaque fichier.

7.2.1 Réponses d'erreur aux commandes MLSx

Beaucoup des réponses 4yz et 5yz définies au paragraphe 4.2 du STD 9, [RFC959] sont possibles en réponse aux commandes MLST et MLSD. En particulier, les erreurs de syntaxe peuvent générer des réponses 500 ou 501. Donner un nom de chemin qui existe mais n'est pas un répertoire comme argument d'une commande MLSD génère une réponse 501. Donner un nom qui n'existe pas, ou pour lequel la permission d'accès (pour obtenir les informations du répertoire comme demandé) n'est pas accordée suscitera une réponse 550. D'autres réponses (530, 553, 503, 504, et toutes les réponses 4yz) sont aussi possibles dans les circonstances appropriées.

7.3 Codage de nom de fichier

Une mise en œuvre de FTP qui prend en charge les commandes MLSx doit être structurée en 8 bits. Cela est nécessaire pour transmettre des noms de fichiers codés en UTF-8. La présente spécification recommande l'utilisation des noms de fichier codés en UTF-8. Les mises en œuvre de FTP DEVRAIENT utiliser l'UTF-8 chaque fois que possible pour faciliter l'interopérabilité maximale.

Les noms de fichiers ne se restreignent pas à l'UTF-8, cependant, le traitement de codages de caractères arbitraires n'est pas spécifié par la présente norme. Les applications sont invitées à traiter les codages non UTF-8 de noms de fichiers comme des séquences d'octets.

Noter que ce codages est sans relation avec le contenu des fichiers, même si le fichier contient des données de caractères.

On trouvera plus d'informations sur le codage des noms de fichiers pour FTP dans "Internationalisation du protocole de transfert de fichier" [RFC2640].

7.3.1 Notes sur le nom de fichier

Le nom de fichier retourné dans la réponse MLST devrait être le même que celui qui a été spécifié dans la commande MLST, ou, si TVFS est pris en charge, un chemin TVFS pleinement qualifié qui désigne le même fichier. Lorsque aucun argument n'a été donné à la commande MLST, le serveur-PI peut inclure un nom de fichier vide dans la réponse, ou il peut fournir un nom qui se réfère au répertoire en cours, si un tel nom est disponible. Lorsque TVFS est pris en charge, un nom de chemin pleinement qualifié du répertoire en cours DEVRAIT être retourné.

Les noms de fichiers retournés dans le résultat d'une commande MLSD DEVRAIENT être des noms non qualifiés au sein du répertoire désigné, ou le répertoire en cours si aucun argument n'était donné. C'est-à-dire que le répertoire désigné dans la commande MLSD NE DEVRAIT PAS apparaître comme un composant des noms de fichiers retournés.

Si le processus de serveur FTP en est capable, et si le fait "type" est retourné, il PEUT retourner, dans la réponse MLSD, une entrée dont le type est "cdir", qui désigne le répertoire d'où le contenu de la liste a été obtenu. Lorsque TVFS est pris en charge, le nom PEUT être le nom de chemin pleinement qualifié du répertoire, ou PEUT être tout autre nom de chemin qui est valide comme référence à ce répertoire à partir du répertoire de travail actuel du serveur FTP. Lorsque plus d'un nom existe, plusieurs de ces entrées peuvent être retournées. Dans un sens, l'entrée "cdir" peut être vue comme un en-tête pour le résultat de MLSD. Cependant, il n'est pas exigé que ce soit la première entrée retournée, et elle peut survenir n'importe où au sein de la liste.

Lorsque TVFS est pris en charge, un usager-PI peut se référer à tout fichier ou répertoire dans la liste en combinant un nom de type "cdir" avec le nom approprié tiré de la liste des répertoires en utilisant la procédure définie au paragraphe 6.2.

Autrement, que TVFS soit pris en charge ou non, l'usager-PI peut produire une commande CWD [RFC959] donnant un nom de type "cdir" à partir de la liste retournée, et à partir de là faire référence aux fichiers retournés dans la réponse MLSD à partir de laquelle le cdir a été obtenu en utilisant les composants de nom de fichier de la liste.

7.4 Format de Faits

Les "faits" pour un fichier dans une réponse à une commande MLSx consistent en informations sur ce fichier. Les faits sont une série de paires mots-clés=valeur dont chacune est suivie par un caractère point-virgule (";"). Un fait individuel ne doit pas contenir un point-virgule dans son nom ou sa valeur. La série complète de faits ne doit pas contenir le caractère espace. Voir la définition ou "RCHAR" au paragraphe 2.1 pour la liste des caractères qui peuvent figurer dans une valeur de fait. Tous ne sont pas applicables à tous les faits.

Un exemple d'une série typique de faits serait :
taille=4161;lang=fr;modifié=19970214165800;créé=19961001124534;type=fichier;x.monfait=foo,bar;

7.5 Faits standard

Le présent document définit un ensemble standard de faits comme suit :

taille	-- Taille en octets
modifié	-- Dernière heure de modification
créé	-- Heure de création
type	-- Type d'entrée
unique	-- Identifiant unique du fichier/répertoire
perm	-- Permissions du fichier, si lecture, écriture, exécution est permis pour l'identifiant de connexion.
lang	-- Langage du nom de fichier selon le registre de [IANA].
media-type	-- Type de support MIME du contenu du fichier selon le registre IANA.
charset	-- Jeu de caractères selon le registre de l'IANA (si ce n'est pas UTF-8)

Les noms de faits sont insensibles à la casse. Taille, taille, TAILLE, et TaiLLe sont le même fait.

D'autres mots-clés spécifiques du système d'exploitation pourraient être spécifiés en utilisant le nom du système d'exploitation de l'IANA comme préfixe (seulement comme exemples) :

OS/2.ea	-- attributs étendus OS/2
MACOS.rf	-- ressources MacIntosh forks
UNIX.mode	-- modes de fichier Unix (permissions)

Les mises en œuvre peuvent définir des mots-clés pour des besoins expérimentaux, ou pour utilisation privée. Tous ces mots-clés DOIVENT commencer par les deux séquences de caractères "x.". Comme les noms de type sont indépendants de la casse, "x." et "X." sont équivalents. Par exemple :

x.ver	-- Informations de version
x.desc	-- Description de fichier
x.type	-- Type de fichier

7.5.1 Type Fait

Le type fait a besoin d'une description particulière. Une partie du problème des pratiques courantes est de décider quand un fichier est un répertoire. Si c'est un répertoire, est-ce le répertoire en cours, un répertoire régulier, ou un répertoire parent ? La spécification de MLST rend cela sans ambiguïté en utilisant le type fait. Le type fait donné spécifie les informations sur l'objet désigné sur la même ligne de la réponse MLST.

Cinq valeurs sont possibles pour le type fait :

fichier	-- une entrée de fichier
cdir	-- le répertoire désigné
pdir	-- un répertoire parent
dir	-- un répertoire or sous-répertoire
OS.name=type	-- un système d'exploitation ou fichier système selon le type de fichier

La syntaxe est définie comme :

type-fait	= type-label "=" type-val
type-label	= "Type"
type-val	= "Fichier" / "cdir" / "pdir" / "dir" / os-type

La valeur du type fait (la "type-val") est une chaîne indépendante de la casse.

7.5.1.1 type=fichier

La présence du fait type=fichier indique que l'entrée citée est un fichier qui contient des données non système. C'est-à-dire qu'il peut être transféré d'un système à un autre de caractéristiques assez différentes, et peut–être avoir encore une signification.

7.5.1.2 type=cdir

Le fait type=cdir indique que l'entrée citée contient un nom de chemin du répertoire dont le contenu est cité. Une entrée de ce type ne sera retournée qu'au titre du résultat d'une commande MLSD lorsque le type fait est inclus, et fournit un nom pour le répertoire cité, et des faits relatifs à ce répertoire. Dans un sens, il peut être vu comme représentant le titre de la

liste, dans un format machine. Il peut apparaître en tout point de la liste ; il n'est pas restreint à une apparition au début, bien que ce soit fréquemment le cas, et il peut survenir plusieurs fois. Il NE DOIT PAS être inclus si le type fait n'est pas inclus, car il n'y aurait pas moyen pour l'usager-PI de distinguer le nom du répertoire d'une entrée dans le répertoire.

Lorsque TVFS est pris en charge par le serveur FTP, ce nom peut être utilisé pour construire des noms de chemin avec lesquels se référer aux fichiers et répertoires retournés dans le même résultat MLSD (voir au paragraphe 6.2). Ces noms de chemins ne sont supposés fonctionner que quand la position du serveur-PI dans l'arborescence de fichier NVFS est la même que sa position lorsque la commande MLSD a été produite, sauf si il en résulte un nom de chemin pleinement qualifié.

Lorsque TVFS n'est pas pris en charge, la seule sémantique définie associée à une entrée "type=cdir" est que, pourvu que le répertoire de travail courant du serveur-PI n'ait pas été changé, un nom de chemin de type "cdir" peut être utilisé comme argument d'une commande CWD, ce qui va faire changer le répertoire en cours du serveur-PI comme le répertoire qui a été cité dans son répertoire de travail courant.

7.5.1.3 type=dir

S'il est présent, l'entrée type=dir donne le nom d'un répertoire. Une telle entrée ne peut normalement pas être transférée d'un système à un autre en utilisant RETR, etc., mais devrait (avec les permissions qui le permettent) être capable d'être l'objet d'une commande MLSD.

7.5.1.4 type=pdir

S'il est présent, ce qui ne va se produire que dans la réponse à une commande MLSD lorsque le type fait est inclus, l'entrée type=pdir représente un nom de chemin du répertoire parent du répertoire cité. En plus d'avoir les propriétés d'un type=dir, une commande CWD qui utilise le nom de chemin tiré de cette entrée devrait changer l'usager en un répertoire parent du répertoire cité. Si le répertoire cité est le répertoire en cours, une commande CDUP peut aussi avoir pour effet de changer le répertoire désigné. Les processus d'usager FTP devraient noter que toutes les réponses ne vont pas inclure ces informations, et que certains systèmes peuvent fournir plusieurs réponses type=pdir.

Lorsque TVFS est pris en charge, un nom "type=pdir" peut être un nom de chemin relatif, ou un nom de chemin pleinement qualifié. Un nom de chemin relatif va se rapporter au répertoire qui est cité, et non au répertoire en cours du serveur-PI à ce moment.

Pour les besoins de cette valeur de type, un "répertoire parent" est tout répertoire dans lequel il y a une entrée de type=dir qui se réfère au répertoire dans lequel a été trouvée l'entité type=pdir. Il n'est donc pas exigé que toutes les entités avec type=pdir se réfèrent au même répertoire. Le fait "unique" (si il est pris en charge et est fourni) peut être utilisé pour déterminer si il y a une relation entre les entrées type=pdir ou non.

7.5.1.5 Types définis par le système

Les types de fichiers qui sont spécifiques d'un certain système d'exploitation, ou fichiers systèmes, peuvent être codés en utilisant les noms de type "OS.". Le format est :

type-os	= "OS." nom-os "=" sorte-d'os
nom-os	= <un nom de système d'exploitation enregistré par l'IANA>
sorte-d'os	= jeton

Le "nom-os" indique le type spécifique de système qui prend en charge le type local particulier. Les types spécifiques d'un OS sont enregistrés par l'IANA en utilisant les procédures spécifiées à la Section 10. Le "sorte-d'os" donne les informations qui dépendent du système comme le type du fichier cité. Les chaînes nom-os et sorte-d'os dans un type-os sont indépendantes de la casse. "OS.unix=bloc" et "OS.Unix=BLOC" représentent le même type (ou le représenterait si un tel type était enregistré.)

Note : Lorsque le système sous-jacent prend en charge un type de fichier qui est essentiellement un pointeur indirect sur un autre fichier, la représentation NVFS de ce type devrait normalement représenter le fichier qu'indique la référence. C'est-à-dire le fichier de base sous-jacent qui apparaît plus d'une fois dans le NVFS, chaque fois avec le fait "unique" (voir le paragraphe qui suit immédiatement) contenant la même valeur, indiquant que le même fichier est représenté par tous ces noms. Les usagers-PI qui transfèrent le fichier ont alors besoin de le transférer une seule fois, et ensuite d'insérer leur propre forme de référence indirecte pour construire des noms de remplacement si ils le désirent, ou peut-être même de copier le fichier local si c'est la seule façon de donner deux noms avec le même contenu. Un fichier qui serait une référence à un autre fichier, si seulement l'autre fichier existe en fait, peut être représenté de toute manière appropriée selon le système d'exploitation, ou pas représenté du tour.

7.5.1.6 Types multiples

Lorsque un fichier est tel qu'il puisse en toute validité, et de façon raisonnable, être traité par le serveur-PI comme étant de plus d'un des types ci-dessus, plusieurs entrées devraient alors être retournées, chacune avec son propre fait "Type" du type

approprié, et contenant chacun le même nom de chemin. Cela peut arriver, par exemple, avec un fichier structuré, qui peut contenir des sous-fichiers, et où le serveur-PI permet que le fichier structuré soit traité comme une unité, ou traité comme un répertoire, permettant que les sous-fichiers soient référencés à l'intérieur de ce fichier. Lorsque cela est fait, le nom de chemin retourné avec chaque entrée DOIT être identique aux autres qui représentent le même fichier.

7.5.2 Fait Unique

Le fait unique est utilisé pour présenter un identifiant univoque pour un fichier ou répertoire dans le NVFS accédé via un processus de serveur FTP. La valeur de ce fait devrait être la même pour tout nombre de noms de chemin qui se réfèrent au même fichier sous-jacent. Le fait devrait avoir des valeurs différentes pour les noms qui font référence à des fichiers distincts. La transposition entre les fichiers et les jetons de fait unique devraient être maintenue, et rester cohérente, pour au moins la durée de vie de la connexion de contrôle entre l'usager-PI et le serveur-PI.

unique-fait = "Unique" "=" jeton

On s'attend à ce que ce fait soit utilisé par les serveurs FTP dont le système hôte permet des choses comme des liaisons symboliques afin que le même fichier puisse être représenté dans plus d'un répertoire sur le serveur. La seule conclusion qui devrait être tirée est que si deux noms différents ont chacun la même valeur pour le fait unique, ils se réfèrent au même objet sous-jacent. La valeur du fait unique (le jeton) devrait être considérée comme une chaîne opaque pour les besoins de comparaison, et est une valeur sensible à la casse. Les jetons "A" et "a" ne représentent pas le même objet sous-jacent.

7.5.3 Fait Modifié

Le fait modifé est utilisé pour déterminer la dernière fois où le contenu du fichier (ou répertoire) indiqué a été modifié. Tout changement de substance du fichier devrait causer l'altération de cette valeur. C'est-à-dire, si un changement est fait à un fichier tel que le résultat d'une commande RETR serait différent, alors la valeur du fait modifié devrait changer. Les usagers-PI ne devraient pas supposer qu'une valeur de fait modifié différente indique que le contenu du fichier sera nécessairement différent de celui qu'il avait lors de la dernière restitution. Certains systèmes peuvent altérer la valeur du fait modifié pour d'autres raisons, bien que ceci soit déconseillé chaque fois que possible. Un fichier peut aussi changer, et revenir alors à son contenu précédent, ce qui sera souvent indiqué comme deux altérations successives de la valeur du fait modifié.

Pour les répertoires, cette valeur devrait changer chaque fois qu'un changement survient au répertoire comme un nom de fichier différent qui serait (ou pourrait) être inclus dans le résultat MLSD de ce répertoire.

fait-modifié = "Modifié" "=" date-de-valeur

7.5.4 Fait Créé

Le fait créé indique quand un fichier, ou répertoire, a été créé. Ce qu'est exactement une "création" n'est pas spécifié ici, et peut varier d'un serveur à l'autre. À peu près tout ce que l'on peut dire sur la valeur retournée est qu'elle ne peut jamais indiquer un moment postérieur à celui du fait modifié

fait-créé = "Créé" "=" date-de-valeur

Note de mise en œuvre : Les mises en œuvre de ce fait sur les systèmes UNIX(TM) devraient noter que le champ unix "stat" "st_ctime" ne donne pas l'heure de création, et que les systèmes de fichiers unix n'enregistrent pas du tout l'heure de création. Les mises en œuvre Unix (et POSIX) ne vont normalement pas inclure ce fait.

7.5.5 Fait Perm

Le fait perm est utilisé pour indiquer les droits d'accès qu'a l'utilisateur FTP en cours sur l'objet cité. Sa valeur est toujours une séquence non ordonnée de caractères alphabétiques.

perm-fait = "Perm" "=" *pvals
pvals = "a" / "c" / "d" / "e" / "f" / "l" / "m" / "p" / "r" / "w"

Dix indicateurs de permission sont actuellement définis. Beaucoup n'ont de signification que lorsque utilisés avec un type d'objet particulier. Les indicateurs ne sont pas sensibles à la casse, "d" et "D" sont le même indicateur.

La permission "a" s'applique aux objets de type=fichier, et indique que la commande APPE (ajouter) peut être appliquée au

fichier cité.

La permission "c" s'applique aux objets du type=dir (et type=pdir, type=cdir). Elle indique que les fichiers peuvent être créés dans le répertoire cité. C'est-à-dire qu'une commande STOU va vraisemblablement réussir, et que les commandes STOR et APPE pourraient réussir si le fichier cité n'existait pas précédemment, mais il est à créer dans l'objet de répertoire qui a lé permission "c". Elle indique aussi que la commande RNTO va vraisemblablement réussir pour les noms dans le répertoire.

La permission "d" s'applique à tous les types. Elle indique que l'objet cité peut être supprimé, c'est-à-dire que la commande RMD peut lui être appliquée si c'est un répertoire, et autrement, que la commande DELE peut lui être appliquée.

La permission "e" s'applique aux types de répertoire. Lorsque elle s'applique à un objet de type=dir, type=cdir, ou type=pdir, elle indique qu'une commande CWD désignant l'objet devrait réussir, et l'usager devrait être capable d'entrer dans le répertoire désigné. Pour le type=pdir, elle indique aussi que la commande CDUP peut réussir (si ce nom de chemin particulier est celui auquel une commande CDUP s'appliquerait).

La permission "f" pour des objets indique que l'objet cité peut être renommé – c'est-à-dire, peut être l'objet d'une commande RNFR.

La permission "l" permission s'applique aux types de fichiers de répertoire, et indique que les commandes de listes, LIST, NLST, et MLSD peuvent être appliquées au répertoire en question.

La permission "m" s'applique aux types de répertoires, et indique que la commande MKD peut être utilisée pour créer un nouveau répertoire au sein du répertoire considéré.

La permission "p" s'applique aux types de répertoires, et indique que les objets dans le répertoire peuvent être supprimés, ou (tordant un peu la désignation) que le répertoire peut être purgé. Noter qu'elle n'indique pas que la commande RMD peut être utilisée pour supprimer le répertoire désigné lui-même, c'est la permission "d" qui indique cela.

La permission "r" s'applique aux objets type=fichier, et pour certains systèmes, peut-être à d'autres types d'objets, et indique que la commande RETR peut être appliquée à cet objet.

La permission "w" s'applique aux objets de type=fichier, et pour certains systèmes, peut-être à d'autres types d'objets, et indique que la commande STOR peut être appliquée à l'objet cité.

Note : L'établissement d'un indicateur de permission n'implique jamais qu'il soit garanti que la commande appropriée va fonctionner – juste comme elle devrait. D'autres limitations spécifiques du système, comme des limitations quant à l'espace disponible pour mémorise les fichiers, peuvent causer un échec de l'opération, alors que les fanions de permission avaient indiqué qu'elle allait vraisemblablement réussir. Les permissions sont seulement une indication.

Note de mise en œuvre : Les permissions sont décrites ici comme elles s'appliquent aux commandes FTP. Elles peuvent ne pas se transposer aisément dans des permissions particulières disponibles sur le système d'exploitation du serveur. Les serveurs sont supposés synthétiser ces bits de permission à partir des informations de permission disponibles dans le système d'exploitation. Par exemple, pour déterminer correctement si le bit de permission "D" devrait être établi sur un répertoire pour un serveur fonctionnant sous le système d'exploitation UNIX(TM), le serveur devrait vérifier que le répertoire désigné est vide, et que l'usager a la permission d'écriture à la fois sur le répertoire considéré et sur son répertoire parent.
Certains systèmes peuvent avoir des permissions plus spécifiques que celles énumérées ici ; de tels systèmes devraient transposer celles-ci en les fanions définis au mieux qu'elles sont capables. D'autres systèmes peuvent avoir seulement de vagues contrôles d'accès. Ils auront généralement juste quelques permutations possibles des fanions de permission, cependant ils devraient tenter de représenter correctement ce qui est permis.

7.5.6 Fait Lang

Le fait lang décrit le langage naturel du nom de fichier pour l'utiliser aux fins d'affichage. Les valeurs utilisées ici devraient être prise dans le registre des langages de l'IANA. Voir dans la [RFC4646] la syntaxe et les procédures qui se rapportent aux étiquettes de langages.

lang-fait = "Lang" "=" jeton

Les mises en œuvre de serveur FTP NE DOIVENT PAS deviner les valeurs de langage. Les valeurs de langue doivent être déterminées d'une façon non ambiguë comme l'étiquetage du langage par un fichier système ou par configuration de

l'utilisateur. Noter que le fait lang ne donne pas d'informations du tout sur le contenu d'un fichier, seulement sur le codage de son nom.

7.5.7 Fait Taille

Le fait taille s'applique aux types de fichiers non répertoire et devrait toujours refléter la taille approximative du fichier. Cela devrait être aussi précis que le serveur peut le faire, sans aller à des longueurs extraordinaires, telles que de lire le fichier entier. La taille est exprimée en unités d'octets de données dans le fichier.

Étant données les limitations de certains systèmes, les mises en œuvre de client FTP doivent comprendre que cette taille peut n'être pas précise et peut changer entre le moment d'une opération MLST et RETR.

Les clients qui ont besoin d'informations de taille très précises pour une raison quelconque devraient utiliser la commande SIZE définie à la Section 4. Le besoin le plus courant pour cette précision est d'être en conjonction avec la commande REST décrite à la Section 5. D'un autre côté, le fait taille devrait être utilisé pour des besoins tels que d'indiquer à un utilisateur humain la taille approximative du fichier à transférer, et peut-être de donner une idée du temps de transfert attendu.

fait-taille = "Taille" "=" 1*CHIFFRE

7.5.8 Le fait Type de support

Le fait Type-de-support représente le type de support selon l'IANA pour le fichier cité, et ne s'applique qu'aux types non répertoire. La liste des valeurs utilisées doit suivre les lignes directrices établies par le registre de l'IANA.

type-de-support = "Type-de-support" "=" <selon les lignes directrices de l'IANA>

Les mises en œuvre de serveur FTP NE DOIVENT PAS inventer des valeurs de type de support. Les valeurs de type de support doivent être déterminées de façon non ambiguë telle qu'un étiquetage de fichier système du type de support ou par configuration par l'utilisateur. Ce fait donne des informations sur le contenu du fichier cité. On devrait donner le type de support principal ainsi que tout sous-type approprié, séparé par une barre oblique "/", comme la tradition en est bien établie.

7.5.9 Le fait Jeu de caractère

Le fait charset (jeu de caractères donne le nom IANA du jeu de caractères, ou un alias, pour les noms de chemin codés dans une réponse MLSx. Le jeu de caractères par défaut est l'UTF-8 sauf mention contraire. Les mises en œuvre de FTP DEVRAIENT utiliser l'UTF-8 si possible pour favoriser l'inter opérabilité maximale. La valeur de ce fait ne s'applique qu'aux noms de chemins, et ne donne pas d'informations sur le contenu du fichier.

charset-type = "Jeu-de-caractère" "=" jeton

7.5.10 Faits exigés

Il n'est pas exigé des serveurs qu'ils prennent en charge un ensemble particulier des faits disponibles. Cependant, les serveurs DEVRAIENT, si c'est possible, prendre au moins en charge les faits type, perm, taille, unique, et modifié.

7.6 Faits dépendants du système et faits locaux

En utilisant un fait qui dépend du système, ou un fait local, un serveur-PI peut communiquer à l'usager-PI des informations sur le fichier cité qui sont particulières au système de fichiers sous-jacent.

7.6.1 Faits dépendants du système

Les noms de faits qui dépendent du système sont étiquetés en ajoutant en préfixe une étiquette qui identifie les informations spécifiques retournées par le nom du système d'exploitation approprié à partir de la liste tenue par l'IANA des noms de systèmes d'exploitation.

La valeur d'un fait dépendant de l'OS peut être tout ce qui est approprié pour porter les informations disponibles. Elle doit cependant être codée comme un "jeton", comme défini au paragraphe 2.1.

Pour permettre une inter opération fiable entre les utilisateurs de faits dépendants du système, l'IANA tiendra un registre

des noms de faits dépendant de systèmes, de leur syntaxe, et de l'interprétation à donner à leurs valeurs. Les enregistrements de faits dépendants de systèmes sont à réaliser conformément aux procédures de la Section 10.

7.6.2 Faits locaux

Les mises en œuvre peuvent aussi rendre disponibles d'autres faits de leur choix. Comme la méthode d'interprétation de telles informations ne sera pas généralement comprise, les serveurs-PI devraient être conscients que les clients vont normalement ignorer tout fait fourni en local. Comme il n'y a pas d'enregistrement des faits définis en local, il est entièrement possible que des serveurs différents utilisent le même nom de fait local pour fournir des informations largement différentes. Les usagers-PI devraient donc y regarder à deux fois avant d'utiliser des informations d'un fait défini en local sans autre assurance spécifique que le fait en question est un de ceux qu'ils comprennent.

Les noms de faits locaux commencent tous par la séquence "X.". Le reste du nom est un "jeton" (voir le paragraphe 2.1). La valeur d'un fait local peut être n'importe quoi, pourvu qu'il soit codé comme "jeton".

7.7 Exemples de MLSx

Les exemples suivants sont tous tirés de dialogues entre des clients et serveurs FTP existants. À cause de cela, toutes les variantes possibles des formats de réponses possibles ne sont pas montrés dans les exemples. Cela ne devrait pas être vu comme une limitation des options d'autres mises en œuvre de serveur. Lorsque les exemples montrent des informations qui dépendent du système d'exploitation, cela doit être considéré comme étant uniquement pour les besoins de la démonstration d'informations spécifique d'OS possibles qui pourraient être définies. Au moment de la rédaction du présent document, aucun fait ou type de fichier spécifique d'OS n'a été défini, les exemples présentés ne devraient en aucun cas être vus comme préférés par rapport à d'autres définitions similaires possibles. Consulter les registres de l'IANA pour déterminer quels types et faits ont été définis. On notera finalement que les exemples présentés sont tirés de mises en œuvre existantes, codées avant l'achèvement du présent document, et qu'il existe une possibilité de différences entre le texte du présent document et les exemples. Dans le cas de telles incohérences, c'est l'exemple qui sera traité comme incorrect.

Dans les exemples présentés, seules les commandes et réponses pertinentes ont été incluses. Cela n'implique pas que d'autres commandes (y compris d'authentification, de modification de répertoire, les commandes PORT ou PASV, ou similaires) ne seraient pas présentes dans une connexion réelle, ou n'étaient pas, en fait, réellement utilisées dans les exemples avant la publication. On notera aussi que les formats présentés sont ceux qui sont transmis entre client et serveur, et non des formats qui seraient normalement rapportés à l'utilisateur du client.

7.7.1 MLST simple

```
C> PWD
S> 257 "/tmp" est le répertoire. en cours.
C> MLst cap60.pl198.tar.gz
S> 250- Listing cap60.pl198.tar.gz
S>  Type=fichier;Taille=1024990;Perm=r; /tmp/cap60.pl198.tar.gz
S> 250Fin
```

Le client demande d'abord qu'on lui dise quel est le répertoire actuel du serveur. C'est simplement pour la clarté de l'exemple. Le client demande ensuite des faits sur un fichier spécifique. Le serveur a retourné la première ligne de réponse de contrôle "250-", suivie par une seule ligne de faits sur le fichier, suivie par la ligne de terminaison "250 ". Le texte sur la ligne de réponse de contrôle et la ligne de terminaison peut être ce que le serveur décide d'envoyer. Remarquer que la ligne de fait est en retrait d'une seule espace. Remarquer aussi qu'il n'y a pas d'espaces dans l'ensemble de faits retourné, jusqu'à la seule espace avant le nom de fichier. Le nom de fichier retourné sur la ligne de faits est un nom de chemin pleinement qualifié du fichier cité. Les faits retournés montrent que la ligne se réfère à un fichier, que ce fichier contient approximativement 1 024 990 octets, soit plus ou moins ce qui peut être transféré si le fichier est restitué, et une quantité différente peut être nécessaire pour mémoriser le fichier dans la mémoire de fichiers du client, et que l'utilisateur connecté a la permission de récupérer le fichier mais pas de faire autre chose qui mérite un intérêt particulier.

7.7.2 MLST d'un répertoire

```
C> PWD
S> 257 "/" est le répertoire. en cours.
C> MLst tmp
S> 250- Listing tmp
S>  Type=dir;Modifié=19981107085215;Perm=el; /tmp
```

S> 250 Fin

Là encore, le PWD est seulement pour les besoins de la démonstration de l'exemple. La ligne de fait MLST montre cette fois que le fichier cité est un répertoire, qu'il a été modifié pour la dernière fois à 08:52:15 le 7 novembre 1998 UTC, et que l'usager a la permission d'entrer dans le répertoire, et de faire la liste de son contenu, mais pas de le modifier d'une quelconque façon. Là encore, le nom de chemin pleinement qualifié du répertoire cité est donné.

7.7.3 MLSD d'un répertoire

C> MLSD tmp
S> 150 connexion BINAIRE ouverte pour MLSD tmp
D> Type=cdir;Modifié=19981107085215;Perm=el; tmp
D> Type=cdir;Modifié=19981107085215;Perm=el; /tmp
D> Type=pdir;Modifié=19990112030508;Perm=el; ..
D> Type=fichier;Taille=25730;Modifié=19940728095854;Perm=; capmux.tar.z
D> Type=fichier;Taille=1830;Modifié=19940916055648;Perm=r; hatch.c
D> Type=fichier;Taille=25624;Modifié=19951003165342;Perm=r; MacIP-02.txt
D> Type=fichier;Taille=2154;Modifié=19950501105033;Perm=r; uar.netbsd.patch
D> Type=fichier;Taille=54757;Modifié=19951105101754;Perm=r; iptnnladev.1.0.sit.hqx
D> Type=fichier;Taille=226546;Modifié=19970515023901;Perm=r; melbcs.tif
D> Type=fichier;Taille=12927;Modifié=19961025135602;Perm=r; tardis.1.6.sit.hqx
D> Type=fichier;Taille=17867;Modifié=19961025135602;Perm=r; timelord.1.4.sit.hqx
D> Type=fichier;Taille=224907;Modifié=19980615100045;Perm=r; uar.1.2.3.sit.hqx
D> Type=fichier;Taille=1024990;Modifié=19980130010322;Perm=r; cap60.pl198.tar.gz
S> 226 MLSD terminé

Dans cet exemple, on remarque qu'il n'y a pas d'espace en tête sur les lignes de fait retournées sur la connexion de données. On remarque aussi que deux lignes de "type=cdir" ont été données. Elles montrent deux noms de remplacement pour le répertoire cité, un de nom de chemin pleinement qualifié, et l'autre avec un nom local relatif au répertoire actuel du serveur lorsque le MLSD a été effectué. Noter que tous les autres noms de fichier dans le résultat sont relatifs au répertoire cité, bien que le serveur pourrait, à son choix, donner un nom de chemin pleinement qualifié pour la ligne "type=pdir". Ce serveur a choisi de ne pas le faire. Les autres fichiers cités présentent un ensemble très ennuyeux de fichiers qui sont présents dans le répertoire cité. Noter qu'il n'y a pas d'ordre particulier pour les énumérer. Il ne sont pas triés par nom de fichier, par taille, ou par heure de modification. Noter aussi que le fait "perm" a une valeur vide pour le fichier "capmux.tar.z", ce qui indique que l'usager connecté n'a aucune permission pour ce fichier. Ce serveur a choisi de présenter les lignes "cdir" et "pdir" avant les lignes qui montrent le contenu du répertoire ; il n'est pas obligé de le faire. Le fait "taille" ne donne pas d'informations significatives pour un répertoire, et il n'est donc pas inclus dans les lignes de fait pour les types de répertoire montrés.

7.7.4 Exemple plus complexe

C> MLst test
S> 250- Listing test
S> Type=dir;Perm=el;Unique=keVO1+ZF4 test
S> 250 End
C> MLSD test
S> 150 BINARY connection open for MLSD test
D> Type=cdir;Perm=el;Unique=keVO1+ZF4; test
D> Type=pdir;Perm=e;Unique=keVO1+d?3; ..
D> Type=OS.unix=slink:/foobar;Perm=;Unique=keVO1+4G4; foobar *(Voir l'errata n° 1500 du 02/09/2008)*
D> Type=OS.unix=chr-13/29;Perm=;Unique=keVO1+5G4; device
D> Type=OS.unix=blk-11/108;Perm=;Unique=keVO1+6G4; block
D> Type=file;Perm=awr;Unique=keVO1+8G4; writable
D> Type=dir;Perm=cpmel;Unique=keVO1+7G4; promiscuous
D> Type=dir;Perm=;Unique=keVO1+1t2; no-exec
D> Type=file;Perm=r;Unique=keVO1+EG4; two words
D> Type=file;Perm=r;Unique=keVO1+IH4; leading space
D> Type=file;Perm=r;Unique=keVO1+1G4; file1
D> Type=dir;Perm=cpmel;Unique=keVO1+7G4; incoming
D> Type=file;Perm=r;Unique=keVO1+1G4; file2
D> Type=file;Perm=r;Unique=keVO1+1G4; file3
D> Type=file;Perm=r;Unique=keVO1+1G4; file4

S> 226 MLSD completed
C> MLSD test/incoming
S> 150 BINARY connection open for MLSD test/incoming
D> Type=cdir;Perm=cpmel;Unique=keVO1+7G4; test/incoming
D> Type=pdir;Perm=el;Unique=keVO1+ZF4; ..
D> Type=file;Perm=awdrf;Unique=keVO1+EH4; bar
D> Type=file;Perm=awdrf;Unique=keVO1+LH4;
D> Type=file;Perm=rf;Unique=keVO1+1G4; file5
D> Type=file;Perm=rf;Unique=keVO1+1G4; file6
D> Type=dir;Perm=cpmdelf;Unique=keVO1+!s2; empty
S> 226 MLSD completed

Pour les besoins de cet exemple, l'ensemble de faits demandé a été modifié pour supprimer les faits "taille" et "modifié", et ajouter le fait "unique". Tout d'abord, les faits sur un nom de fichier ont été obtenus via MLST. Noter qu'aucun nom de chemin pleinement qualifié n'a été donné cette fois ci. C'est parce que le serveur n'était pas capable de déterminer ces informations. Puis, ayant déterminé que le nom de fichier représente un répertoire, la liste de ce répertoire a été faite. Cette liste ne montre pas non plus de nom de chemin pleinement qualifié, pour la même raison, et a donc une seule ligne "type=cdir". Ce répertoire (qui a été créé spécialement à cette fin) contient plusieurs fichiers intéressants. Il y en a avec des types de fichier qui dépendent de l'OS, plusieurs sous-répertoires, et plusieurs fichiers ordinaires.

On ne peut pas dire grand chose ici sur les types de fichier dépendants de l'OS, car aucune des informations qui y sont montrées ne peut être traitée comme étant plus qu'une simple possibilité. On peut voir cependant que le type d'OS du serveur est "unix", qui est un des types d'OS du registre IANA des noms de système d'exploitation.

Des trois répertoires cités, "no-exec" n'a pas de permission accordée à cet usager pour accéder du tout. De la valeur de fait "Unique" on peut déduire que "promiscuous" et "incoming" représentent en fait le même répertoire. Ses permissions montrent que l'usager connecté a la permission de faire essentiellement tout sauf supprimer le répertoire. Ce répertoire a été cité ultérieurement. Il se trouve que le répertoire ne peut pas être supprimé parce qu'il n'est pas vide.

Des fichiers normaux cités, deux contiennent des espaces dans leur nom. Le fichier nommé " leading space" contient en fait deux espaces dans son nom, un avant le "l" et un entre le "g" et le "s". Les deux espaces qui séparent les faits de la partie visible du nom de chemin rendent cela clair. Le fichier "writable" a les bits de permission "a" et "w" établis, et par conséquent l'usager connecté devrait être capable d'appliquer STOR ou APPE à ce fichier.

Les quatre autres noms de fichiers, "file1", "file2", "file3", et "file4" représentent tous le même fichier sous-jacent, comme on peut le voir d'après les valeurs des faits "unique" de chacun. Il se trouve que "file1" et "file2" sont des liens Unix "hard", et que "file3" et "file4" sont des liens "soft" ou "symbolic" des deux premiers. Aucune de ces informations n'est disponible via des faits MLST standard ; il est suffisant pour les besoins de FTP de noter que tous représentent le même fichier, et que les mêmes données seraient collectées sans considération de la provenance de celles qui sont restituées, et que tous seraient simultanément modifiés si des données étaient mémorisées dans l'un d'entre eux.

Finalement, le sous-répertoire "incoming" est cité. Comme "promiscuous" est le même répertoire il n'y a pas lieu de le citer non plus. Dans ce répertoire, les fichiers "file5" et "file6" représentent encore plus de noms pour le fichier "file1" qu'on a vu précédemment. Remarquer l'entrée entre cela pour "bar" et "file5". Bien qu'il ne soit pas possible de le représenter facilement dans le présent document, cela montre un fichier avec un nom comprenant exactement trois espaces (" "). Un client n'aura cependant pas de difficulté à déterminer ce nom à partir du résultat qui lui est présenté. Le répertoire "empty" est, comme son nom l'indique, vide, bien que ce ne soit pas montré ici. Il peut, cependant, être supprimé, comme le fichier "bar" et le fichier dont le nom est trois espaces. Tous les fichiers qui résident dans ce répertoire peuvent être renommés. C'est une conséquence de la sémantique de répertoire UNIX qui les contient qu'ils soient modifiables.

7.7.5 Informations d'heure plus précises

C> MLst file1
S> 250- Listing file1
S> Type=file;Modify=19990929003355.237; file1
S> 250 End

Dans cet exemple, le serveur FTP indique que "file1" a été modifié 237 millisecondes après 00:33:55 UTC le 29 septembre 1999.

7.7.6 Un serveur différent

C> MLST
S> 250-Begin
S> type=dir;unique=AQkAAAAAAAABCAAA; /
S> 250 End.
C> MLSD
S> 150 Opening ASCII mode connexion de données for MLS.
D> type=cdir;unique=AQkAAAAAAAABCAAA; /
D> type=dir;unique=AQkAAAAAAAABEAAA; bin
D> type=dir;unique=AQkAAAAAAAABGAAA; etc
D> type=dir;unique=AQkAAAAAAAAB8AwA; halflife
D> type=dir;unique=AQkAAAAAAAABoAAA; incoming
D> type=dir;unique=AQkAAAAAAAABIAAA; lib
D> type=dir;unique=AQkAAAAAAAABWAEA; linux
D> type=dir;unique=AQkAAAAAAAABKAEA; ncftpd
D> type=dir;unique=AQkAAAAAAAABGAEA; outbox
D> type=dir;unique=AQkAAAAAAAABuAAA; quake2
D> type=dir;unique=AQkAAAAAAAABQAEA; winstuff
S> 226 Listing completed.
C> MLSD linux
S> 150 Opening ASCII mode connexion de données for MLS.
D> type=cdir;unique=AQkAAAAAAAABWAEA; /linux
D> type=pdir;unique=AQkAAAAAAAABCAAA; /
D> type=dir;unique=AQkAAAAAAAABeAEA; firewall
D> type=file;size=12;unique=AQkAAAAAAAACWAEA; helo_world
D> type=dir;unique=AQkAAAAAAAABYAEA; kernel
D> type=dir;unique=AQkAAAAAAAABmAEA; scripts
D> type=dir;unique=AQkAAAAAAAABkAEA; security
S> 226 Listing completed.
C> MLSD linux/kernel
S> 150 Opening ASCII mode connexion de données for MLS.
D> type=cdir;unique=AQkAAAAAAAABYAEA; /linux/kernel
D> type=pdir;unique=AQkAAAAAAAABWAEA; /linux
D> type=file;size=6704;unique=AQkAAAAAAAADYAEA; k.config
D> type=file;size=7269221;unique=AQkAAAAAAAACYAEA; linux-2.0.36.tar.gz
D> type=file;size=12514594;unique=AQkAAAAAAAAEYAEA; linux-2.1.130.tar.gz
S> 226 Listing completed.

Noter que ce serveur retourne sa valeur de fait "unique" dans un format assez différent. Il retourne aussi des noms de chemin pleinement qualifiés pour l'entrée "pdir".

7.7.7 Quelques fichiers IANA

C> MLSD
S> 150 BINARY connection open for MLSD .
D> Type=cdir;Modify=19990219183438; /iana/assignments
D> Type=pdir;Modify=19990112030453; ..
D> Type=dir;Modify=19990219073522; media-types
D> Type=dir;Modify=19990112033515; character-set-info
D> Type=dir;Modify=19990112033529; languages
D> Type=file;Size=44242;Modify=19990217230400; character-sets
D> Type=file;Size=1947;Modify=19990209215600; operating-system-names
S> 226 MLSD completed
C> MLSD media-types
S> 150 BINARY connection open for MLSD media-types
D> Type=cdir;Modify=19990219073522; media-types
D> Type=cdir;Modify=19990219073522; /iana/assignments/media-types
D> Type=pdir;Modify=19990219183438; ..
D> Type=dir;Modify=19990112033045; text
D> Type=dir;Modify=19990219183442; image
D> Type=dir;Modify=19990112033216; multipart
D> Type=dir;Modify=19990112033254; video

D> Type=file;Size=30249;Modify=19990218032700; media-types
S> 226 MLSD completed
C> MLSD character-set-info
S> 150 BINARY connection open for MLSD character-set-info
D> Type=cdir;Modify=19990112033515; character-set-info
D> Type=cdir;Modify=19990112033515; /iana/assignments/character-set-info
D> Type=pdir;Modify=19990219183438; ..
D> Type=file;Size=1234;Modify=19980903020400; windows-1251
D> Type=file;Size=4557;Modify=19980922001400; tis-620
D> Type=file;Size=801;Modify=19970324130000; ibm775
D> Type=file;Size=552;Modify=19970320130000; ibm866
D> Type=file;Size=922;Modify=19960505140000; windows-1258
S> 226 MLSD completed
C> MLSD languages
S> 150 BINARY connection open for MLSD languages
D> Type=cdir;Modify=19990112033529; languages
D> Type=cdir;Modify=19990112033529; /iana/assignments/languages
D> Type=pdir;Modify=19990219183438; ..
D> Type=file;Size=2391;Modify=19980309130000; default
D> Type=file;Size=943;Modify=19980309130000; tags
D> Type=file;Size=870;Modify=19971026130000; navajo
D> Type=file;Size=699;Modify=19950911140000; no-bok
S> 226 MLSD completed
C> PWD
S> 257 "/iana/assignments" is current répertoire.

Cet exemple montre des fichiers tenus par l'IANA qui sont pertinents pour la présente spécification en format MLSD. Noter que ces listes ont été éditées en supprimant de nombreuses entrées, les listes réelles sont beaucoup plus longues.

7.7.8 Essai de résistance à la dépendance à la casse

L'exemple suivant est destiné à éclairer certains cas où des chaînes dépendantes de la casse sont permises dans les commandes MLSx, et où des chaînes indépendantes de la casse sont exigées.

Noter d'abord que la commande "MLSD", montrée ici par "MlsD" est indépendante de la casse. Les clients peuvent produire cette commande dans n'importe quelle casse, ou combinaison de casses, qu'ils désirent. C'est le cas de toutes les commandes FTP.

C> MlsD
S> 150 BINARY connection open for MLSD .
D> Type=pdir;Modify=19990929011228;Perm=el;Unique=keVO1+ZF4; ..
D> Type=file;Size=4096;Modify=19990929011440;Perm=r;Unique=keVO1+Bd8; FILE2
D> Type=file;Size=4096;Modify=19990929011440;Perm=r;Unique=keVO1+aG8; file3
D> Type=file;Size=4096;Modify=19990929011440;Perm=r;Unique=keVO1+ag8; FILE3
D> Type=file;Size=4096;Modify=19990929011440;Perm=r;Unique=keVO1+bD8; file1
D> Type=file;Size=4096;Modify=19990929011440;Perm=r;Unique=keVO1+bD8; file2
D> Type=file;Size=4096;Modify=19990929011440;Perm=r;Unique=keVO1+Ag8; File3
D> Type=file;Size=4096;Modify=19990929011440;Perm=r;Unique=keVO1+bD8; File1
D> Type=file;Size=4096;Modify=19990929011440;Perm=r;Unique=keVO1+Bd8; File2
D> Type=file;Size=4096;Modify=19990929011440;Perm=r;Unique=keVO1+bd8; FILE1
S> 226 MLSD completed

Remarquer ensuite les étiquettes des faits. Ce sont aussi des chaînes indépendantes de la casse ; il est permis au serveur FTP de les retourner dans toute casse désirée. Les usagers FTP doivent être prêts à traiter toute casse, bien qu'il puissent faire cela en transposant les étiquettes en une casse commune si ils le désirent.

Ensuite, on remarque qu'il y a neuf objets de "type" fichier retournés. Dans un NVFS indépendant de la casse ils représenteraient trois noms de fichier différents, "file1", "file2", et "file3". Avec un NVFS dépendant de la casse, tous les neuf représentent des noms de fichier différents. L'un et l'autre sont possibles ; les serveurs FTP peuvent mettre en œuvre un NVFS dépendant ou indépendant de la casse. Les usagers FTP doivent permettre un choix de fichiers dépendant de la casse à manipuler sur le serveur.

Enfin, on remarque que la valeur du fait "unique" dépend de la casse. Dans l'exemple présenté, "file1", "File1", et "file2" ont tous la même valeur de fait "unique" "keVO1+bD8", et représentent donc tous le même fichier sous-jacent. D'un autre côté, "FILE1" a une valeur différente de fait "unique" ("keVO1+bd8") et représente donc un fichier différent. De même, "FILE2" et "File2" sont deux noms pour le même fichier sous-jacent, tandis que "file3", "File3" et "FILE3" représentent tous des fichiers sous-jacents différents.

Que les tailles approximatives (fait "size") et les heures de dernière modification (fait "modify") soient les mêmes dans tous les cas peut n'être qu'une simple coïncidence.

Il n'est pas suggéré que les opérateurs de serveurs FTP créent un NVFS qui sollicite à ce point le protocole ; cependant, les mises en œuvre d'usager comme de serveur doivent être prêtes à traiter des exemples aussi extrêmes.

7.7.9 Exemple d'un autre serveur

```
C> MlsD
S> 150 File Listing Follows in IMAGE / Binary mode.
D> type=cdir;modify=19990426150227;perm=el; /MISC
D> type=pdir;modify=19791231130000;perm=el; /
D> type=dir;modify=19990426150227;perm=el; CVS
D> type=dir;modify=19990426150228;perm=el; SRC
S> 226 Transfer finished successfully.
C> MlsD src
S> 150 File Listing Follows in IMAGE / Binary mode.
D> type=cdir;modify=19990426150228;perm=el; /MISC/src
D> type=pdir;modify=19990426150227;perm=el; /MISC
D> type=dir;modify=19990426150228;perm=el; CVS
D> type=dir;modify=19990426150228;perm=el; INSTALL
D> type=dir;modify=19990426150230;perm=el; INSTALLI
D> type=dir;modify=19990426150230;perm=el; TREES
S> 226 Transfer finished successfully.
C> MlsD src/install
S> 150 File Listing Follows in IMAGE / Binary mode.
D> type=cdir;modify=19990426150228;perm=el; /MISC/src/install
D> type=pdir;modify=19990426150228;perm=el; /MISC/src
D> type=file;modify=19990406234304;perm=r;size=20059; BOOTPC.C
D> type=file;modify=19980401170153;perm=r;size=278; BOOTPC.H
D> type=file;modify=19990413153736;perm=r;size=54220; BOOTPC.O
D> type=file;modify=19990223044003;perm=r;size=3389; CDROM.C
D> type=file;modify=19990413153739;perm=r;size=30192; CDROM.O
D> type=file;modify=19981119155324;perm=r;size=1055; CHANGELO
D> type=file;modify=19981204171040;perm=r;size=8297; COMMANDS.C
D> type=file;modify=19980508041749;perm=r;size=580; COMMANDS.H
                                ...
D> type=file;modify=19990419052351;perm=r;size=54264; URLMETHO.O
D> type=file;modify=19980218161629;perm=r;size=993; WINDOWS.C
D> type=file;modify=19970912154859;perm=r;size=146; WINDOWS.H
D> type=file;modify=19990413153731;perm=r;size=16812; WINDOWS.O
D> type=file;modify=19990322174959;perm=r;size=129; _CVSIGNO
D> type=file;modify=19990413153640;perm=r;size=82536; _DEPEND
S> 226 Transfer finished successfully.
C> MLst src/install/windows.c
S> 250-Listing src/install/windows.c
S>  type=file;perm=r;size=993; /misc/src/install/windows.c
S> 250 End
S> ftp> mlst SRC/INSTALL/WINDOWS.C
C> MLst SRC/INSTALL/WINDOWS.C
S> 250-Listing SRC/INSTALL/WINDOWS.C
S>  type=file;perm=r;size=993; /misc/SRC/INSTALL/WINDOWS.C
S> 250 End
```

Noter que ce serveur donne des noms de chemin pleinement qualifiés pour les entrées "pdir" et "cdir" dans les listes MLSD. Remarquer aussi que ce serveur trie, bien qu'il n'y soit pas obligé, ses résultats de listes de répertoire. Ce peut être, bien sûr, un artifice des mécanismes d'accès du système de fichiers sous-jacent. Remarquer enfin que le NVFS pris en charge

par ce serveur, à l'opposé des précédents, met en œuvre ses noms de chemin de façon indépendante de la casse. Le serveur semble retourner les fichiers en utilisant la casse dans laquelle ils ont été demandés, lorsque le nom a été envoyé par le client, et autrement, utilise un algorithme connu de lui seul pour choisir la casse des noms qu'il retourne.

7.7.10 Serveur qui s'écoute lui-même

```
C> MLst f
S> 250-MLST f
S>  Type=dir;Modify=20000710052229;Unique=AAD/AAAABIA; f
S> 250 End
C> CWD f
S> 250 CWD command successful.
C> MLSD
S> 150 Opening ASCII mode data connection for 'MLSD'.
D> Type=cdir;Unique=AAD/AAAABIA; .
D> Type=pdir;Unique=AAD/AAAAAAI; ..
D> Type=file;Size=987;Unique=AAD/AAAABIE; Makefile
D> Type=file;Size=20148;Unique=AAD/AAAABII; conf.c
D> Type=file;Size=11111;Unique=AAD/AAAABIM; extern.h
D> Type=file;Size=38721;Unique=AAD/AAAABIQ; ftpcmd.y
D> Type=file;Size=17922;Unique=AAD/AAAABIU; ftpd.8
D> Type=file;Size=60732;Unique=AAD/AAAABIY; ftpd.c
D> Type=file;Size=3127;Unique=AAD/AAAABIc; logwtmp.c
D> Type=file;Size=2294;Unique=AAD/AAAABIg; pathnames.h
D> Type=file;Size=7605;Unique=AAD/AAAABIk; popen.c
D> Type=file;Size=9951;Unique=AAD/AAAABIo; ftpd.conf.5
D> Type=file;Size=5023;Unique=AAD/AAAABIs; ftpusers.5
D> Type=file;Size=3547;Unique=AAD/AAAABIw; logutmp.c
D> Type=file;Size=2064;Unique=AAD/AAAABI0; version.h
D> Type=file;Size=20420;Unique=AAD/AAAAAAM; cmds.c
D> Type=file;Size=15864;Unique=AAD/AAAAAAg; ls.c
D> Type=file;Size=2898;Unique=AAD/AAAAAAk; ls.h
D> Type=file;Size=2769;Unique=AAD/AAAAAAo; lsextern.h
D> Type=file;Size=2042;Unique=AAD/AAAAAAs; stat_flags.h
D> Type=file;Size=5708;Unique=AAD/AAAAAAw; cmp.c
D> Type=file;Size=9280;Unique=AAD/AAAAAA0; print.c
D> Type=file;Size=4657;Unique=AAD/AAAAAA4; stat_flags.c
D> Type=file;Size=2664;Unique=AAD/AAAAAA8; util.c
D> Type=file;Size=10383;Unique=AAD/AAAABJ0; ftpd.conf.cat5
D> Type=file;Size=3631;Unique=AAD/AAAABJ4; ftpusers.cat5
D> Type=file;Size=17729;Unique=AAD/AAAABJ8; ftpd.cat8
S> 226 MLSD complete.
```

Cet exemples montre encore une autre mise en œuvre de serveur, qui donne une liste de son propre code source. Noter que cette mise en œuvre ne comporte par le nom de chemin pleinement qualifié dans ses entrées "cdir" et "pdir", ni dans le résultat de "MLST". Noter aussi que les faits demandés ont été modifiés entre les commandes "MLST" et "MLSD", bien que cet échange n'ait pas été montré ici.

7.7.11 Serveur avec une différence

```
C> PASV
S> 227 Entering Passive Mode (127,0,0,1,255,46)
C> MLSD
S> 150 I tink I tee a trisector tree
D> Type=file;Unique=aaaaafUYqaaa;Perm=rf;Size=15741; x
D> Type=cdir;Unique=aaaaacUYqaaa;Perm=cpmel; /
D> Type=file;Unique=aaaaajUYqaaa;Perm=rf;Size=5760; x4
D> Type=dir;Unique=aaabcaUYqaaa;Perm=elf; sub
D> Type=file;Unique=aaaaagUYqaaa;Perm=rf;Size=8043; x1
D> Type=dir;Unique=aaab8aUYqaaa;Perm=cpmelf; files
D> Type=file;Unique=aaaaahUYqaaa;Perm=rf;Size=4983; x2
D> Type=file;Unique=aaaaaiUYqaaa;Perm=rf;Size=6854; x3
```

S> 226 That's all folks...
C> CWD sub
S> 250 CWD command successful.
C> PWD
S> 257 "/sub" is current directory.
C> PASV
S> 227 Entering Passive Mode (127,0,0,1,255,44)
C> MLSD
S> 150 I tink I tee a trisector tree
D> Type=dir;Unique=aaabceUYqaaa;Perm=elf; dir
D> Type=file;Unique=aaabcbUYqaaa;Perm=rf;Size=0; y1
D> Type=file;Unique=aaabccUYqaaa;Perm=rf;Size=0; y2
D> Type=file;Unique=aaabcdUYqaaa;Perm=rf;Size=0; y3
D> Type=pdir;Unique=aaaaacUYqaaa;Perm=cpmel; /
D> Type=pdir;Unique=aaaaacUYqaaa;Perm=cpmel; ..
D> Type=cdir;Unique=aaabcaUYqaaa;Perm=el; /sub
S> 226 That's all folks...
C> PASV
S> 227 Entering Passive Mode (127,0,0,1,255,42)
C> MLSD dir
S> 150 I tink I tee a trisector tree
D> Type=pdir;Unique=aaabcaUYqaaa;Perm=el; /sub
D> Type=pdir;Unique=aaabcaUYqaaa;Perm=el; ..
D> Type=file;Unique=aaab8cUYqaaa;Perm=r;Size=15039; mlst.c
D> Type=dir;Unique=aaabcfUYqaaa;Perm=el; ect
D> Type=cdir;Unique=aaabceUYqaaa;Perm=el; dir
D> Type=cdir;Unique=aaabceUYqaaa;Perm=el; /sub/dir
D> Type=dir;Unique=aaabchUYqaaa;Perm=el; misc
D> Type=file;Unique=aaab8bUYqaaa;Perm=r;Size=34589; ftpd.c
S> 226 That's all folks...
C> CWD dir/ect
S> 250 CWD command successful.
C> PWD
S> 257 "/sub/dir/ect" is current directory.
C> PASV
S> 227 Entering Passive Mode (127,0,0,1,255,40)
C> MLSD
S> 150 I tink I tee a trisector tree
D> Type=dir;Unique=aaabcgUYqaaa;Perm=el; ory
D> Type=pdir;Unique=aaabceUYqaaa;Perm=el; /sub/dir
D> Type=pdir;Unique=aaabceUYqaaa;Perm=el; ..
D> Type=cdir;Unique=aaabcfUYqaaa;Perm=el; /sub/dir/ect
S> 226 That's all folks...
C> CWD /files
S> 250 CWD command successful.
C> PASV
S> 227 Entering Passive Mode (127,0,0,1,255,36)
C> MLSD
S> 150 I tink I tee a trisector tree
D> Type=cdir;Unique=aaab8aUYqaaa;Perm=cpmel; /files
D> Type=pdir;Unique=aaaaacUYqaaa;Perm=cpmel; /
D> Type=pdir;Unique=aaaaacUYqaaa;Perm=cpmel; ..
D> Type=file;Unique=aaab8cUYqaaa;Perm=rf;Size=15039; mlst.c
D> Type=file;Unique=aaab8bUYqaaa;Perm=rf;Size=34589; ftpd.c
S> 226 That's all folks...
C> RNFR mlst.c
S> 350 File exists, ready for destination name
C> RNTO list.c
S> 250 RNTO command successful.
C> PASV
S> 227 Entering Passive Mode (127,0,0,1,255,34)
C> MLSD
S> 150 I tink I tee a trisector tree

D> Type=file;Unique=aaab8cUYqaaa;Perm=rf;Size=15039; list.c
D> Type=pdir;Unique=aaaaacUYqaaa;Perm=cpmel; /
D> Type=pdir;Unique=aaaaacUYqaaa;Perm=cpmel; ..
D> Type=file;Unique=aaab8bUYqaaa;Perm=rf;Size=34589; ftpd.c
D> Type=cdir;Unique=aaab8aUYqaaa;Perm=cpmel; /files
S> 226 That's all folks...

Le serveur qu'on montre ici retourne ses listes de répertoires dans un ordre assez aléatoire, et semble même modifier l'ordre du répertoire lorsque son contenu change – peut-être que la structure sous-jacente du répertoire se fonde sur une certaine forme de hachage. Noter que les entrées "pdir" et "cdir" sont entremêlées avec d'autres entrées dans le répertoire. Noter aussi que ce serveur n'affiche pas d'entrée "pdir" lorsque il énumère le contenu du répertoire racine de la mémorisation virtuelle des fichiers ; cependant, il comporte pourtant plusieurs entrées "cdir" et "pdir" lorsque il s'y sent enclin. Le serveur utilise aussi des messages d'une répugnante "familiarité".

7.8 Réponses FEAT pour MLSx

Lorsque il répond à la commande FEAT, un processus de serveur FTP qui prend en charge MLST et MLSD, plus l'internationalisation des noms de chemins, DOIT indiquer que cette prise en charge existe. Il le fait en incluant une ligne de caractéristiques MLST. Tout en indiquant le soutien de base, la ligne de caractéristiques MLST indique quels faits MLST sont disponibles à partir du serveur, et lesquels d'entre eux seront retournés si aucune autre commande "OPTS MLST" ultérieure n'est envoyée.

```
mlst-feat    = SP "MLST" [SP factlist] CRLF
factlist     = 1*( factname ["*"] ";" )
```

L'espace initiale montrée dans la réponse mlst-feat est celle exigée par la commande FEAT, deux espaces n'est pas permis. Si une factlist n'est pas donnée, le processus de serveur FTP indique alors qu'il prend en charge MLST, mais ne met en œuvre aucun fait. Seuls des noms de chemin peuvent être retournés. Cela serait une mise en œuvre MLST minimale, et peu utile pour la plupart des utilisations pratiques. Lorsque la factlist est présente, les factnames inclus indiquent les faits pris en charge par le serveur. Lorsque l'astérisque facultative apparaît après un factname, ce fait sera inclus dans les réponses au format MLST, jusqu'à ce qu'un "OPTS MLST" soit donné pour altérer la liste des faits retournée. Après cela, les commandes FEAT suivantes vont retourner l'astérisque pour montrer les faits choisis par le plus récent "OPTS MLST".

Noter qu'il n'y a pas de résultat FEAT distinct pour MLSD. La présence de la caractéristique MLST indique que MLST et MLSD sont tous deux pris en charge.

7.8.1 Exemples

C> Feat
S> 211- Features supported
S> REST STREAM
S> MDTM
S> SIZE
S> TVFS
S> UTF8
S> MLST Type*;Size*;Modify*;Perm*;Unique*;UNIX.mode;UNIX.chgd;X.hidden;
S> 211 End

En plus de certaines caractéristiques non pertinentes ici, ce serveur indique qu'il prend en charge MLST y compris plusieurs faits standard, mais pas toutes, qu'il va envoyer par défaut. Il prend aussi en charge deux faits dépendants de l'OS, et un fait défini en local. Ces trois derniers doivent être expressément demandés par le client pour que ce serveur les fournisse.

C> Feat
S> 211-Extensions supported:
S> CLNT
S> MDTM
S> MLST type*;size*;modify*;UNIX.mode*;UNIX.owner;UNIX.group;unique;
S> PASV
S> REST STREAM
S> SIZE
S> TVFS

S> Compliance Level: 19981201 (IETF mlst-05)
S> 211 End.

Là encore, en plus de certaines caractéristiques non pertinentes pour notre propos, ce serveur indique qu'il prend en charge MLST, quatre des faits standard, dont un ("unique") n'est pas activé par défaut, et plusieurs faits dépendants de l'OS, dont un est fourni par défaut par le serveur. Ce serveur prend en fait en charge plus de faits dépendants de l'OS. Les autres ont été supprimés pour les besoins du présent document pour se conformer aux contraintes de formatage du document.

C> FEAT
S> 211-Features supported
S> MDTM
S> MLST Type*;Size*;Modify*;Perm;Unique*;
S> REST STREAM
S> SIZE
S> TVFS
S> 211 End

Ce serveur a sagement choisi de ne pas mettre en œuvre de fait dépendant de l'OS. Au moment de la rédaction du présent document, de tels faits n'ont pas encore été définis (en utilisant les mécanismes du paragraphe 10.1) et donc une prise en charge raisonnée de tels faits serait au mieux difficile. Tous les faits pris en charge par ce serveur sauf un sont activés par défaut.

7.9 Paramètres OPTS pour MLST

Pour les commandes MLSx, le client FTP peut spécifier une liste de faits dont il souhaite qu'ils soient retournés dans toutes les commandes MLSx suivantes jusqu'à ce qu'une autre commande OPTS MLST soit envoyée. Le format spécifié est :

mlst-opts = "OPTS" SP "MLST" [SP 1*(factname ";")]

En envoyant la commande "OPTS MLST", le client demande au serveur de n'inclure que les faits énumérés comme arguments à la commande dans les résultats suivants des commandes MLSx. Les faits non inclus dans la commande "OPTS MLST" NE DOIVENT PAS être retournés par le serveur. Les faits qui sont inclus devraient être retournés pour chaque entrée retournée à partir de la commande MLSx à laquelle ils s'appliquent. Les faits demandés qui ne sont pas pris en charge, ou qui sont inappropriés pour le fichier ou répertoire cité devraient simplement être omis du résultat MLSx. Ceci n'est pas une erreur. Noter que lorsque aucun argument factname n'est présent, le client demande que seuls les noms de fichier soient retournés. Dans ce cas, et dans tout autre cas où aucun fait n'est inclus dans le résultat, l'espace qui sépare les noms de fait et leurs valeurs des noms de fichiers est toujours exigée. C'est-à-dire que le premier caractère de la ligne de résultat sera une espace, (ou deux caractères seront des espaces lorsque la ligne est retournée sur la connexion de contrôle) et le nom de fichier va commencer immédiatement après cela.

Les clients devraient noter que générer des valeurs pour certains faits peut être possible, mais très coûteux, pour certains serveurs. Il est généralement acceptable de restituer tous les faits qu'offre le serveur comme ensemble par défaut avant que toute commande "OPTS MLST" ait été donnée, cependant les clients devraient faire très attention avant de demander un fait qui n'est pas dans cet ensemble. C'est-à-dire que bien que d'autres faits puissent être disponibles depuis le serveur, les clients devraient s'abstenir de les demander sauf si ils ont une exigence de fonctionnement particulière pour ces informations, qui devrait être plus significative que peut-être de simplement améliorer les informations affichées chez l'utilisateur final.

Noter qu'il n'y a pas de commande "OPTS MLSD" ; les noms de fait établis avec la commande "OPTS MLST" s'appliquent aux deux commandes MLST et MLSD.

Les serveurs ne sont pas obligés d'accepter les commandes "OPTS MLST" avant l'authentification de l'usager-PI, mais ils peuvent choisir de les permettre.

7.9.1 Réponse OPTS MLST

Le "message-de-réponse" de la [RFC2389] à une commande OPTS MLST réussie a la syntaxe suivante :

mlst-opt-resp = "MLST OPTS" [SP 1*(factname ";")]

Cela définit le "message-de-réponse" comme utilisé dans le message "opts-good" de la [RFC2389].

Les faits nommés dans la réponse sont ceux que le serveur va maintenant inclure dans la réponse MLST (et MLSD) après le traitement de la commande "OPTS MLST". Tout fait de la demande qui n'est pas pris en charge par le serveur sera omis de ce message de réponse. Si aucun fait n'est inclus, la liste de faits sera vide. Noter que la liste des faits retournés sera la même que celle des faits marqués par une astérisque ("*") en fin de mot dans une réponse de commande FEAT suivante. Il n'est pas exigé que l'ordre des faits retournés soit le même que celui dans lequel ils étaient demandés, ou celui dans lequel ils seront énumérés dans une réponse de commande FEAT, ou celui dans lequel les faits sont retournés dans les réponses MLST. La chaîne fixe "MLST OPTS" dans la réponse peut être retournée dans n'importe quelle casse, ou mélange de casses.

7.9.2 Exemples

C> Feat
S> 211- Features supported
S> MLST Type*;Size;Modify*;Perm;Unique;UNIX.mode;UNIX.chgd;X.hidden;
S> 211 End
C> OptS Mlst Type;UNIX.mode;Perm;
S> 200 MLST OPTS Type;Perm;UNIX.mode;
C> Feat
S> 211- Features supported
S> MLST Type*;Size;Modify;Perm*;Unique;UNIX.mode*;UNIX.chgd;X.hidden;
S> 211 End
C> opts MLst lang;type;charset;create;
S> 200 MLST OPTS Type;
C> Feat
S> 211- Features supported
S> MLST Type*;Size;Modify;Perm;Unique;UNIX.mode;UNIX.chgd;X.hidden;
S> 211 End
C> OPTS mlst size;frogs;
S> 200 MLST OPTS Size;
C> Feat
S> 211- Features supported
S> MLST Type;Size*;Modify;Perm;Unique;UNIX.mode;UNIX.chgd;X.hidden;
S> 211 End
C> opts MLst unique type;
S> 501 Invalid MLST options
C> Feat
S> 211- Features supported
S> MLST Type;Size*;Modify;Perm;Unique;UNIX.mode;UNIX.chgd;X.hidden;
S> 211 End

Pour les besoins de cet exemple, les caractéristiques autres que de MLST ont été supprimées de la sortie afin d'éviter d'encombrer. L'exemple montre le résultat de caractéristique initiale par défaut pour MLST. Les faits demandés sont alors changés par le client. Le premier changement montre des faits qui sont disponibles à partir du serveur choisi. Les résultats FEAT suivants montrent les caractéristiques altérées retournées. Le client tente alors de choisir des caractéristiques standard que le serveur ne prend pas en charge. Ce n'est pas une erreur, cependant le serveur ignore simplement les demandes de caractéristiques non prises en charge, comme le montre le résultat FEAT qui suit. Puis, le client tente de demander une caractéristique non standard, et non prise en charge. Le serveur l'ignore, et choisit seulement les caractéristiques demandées qu'il prend en charge. Enfin, le client envoie une demande contenant une erreur de syntaxe (des espaces ne peuvent pas apparaître dans la factlist). Le serveur FTP envoie une réponse d'erreur et ignore complètement la demande, laissant l'ensemble de faits comme il était antérieurement.

Noter que dans tous les cas, sauf celui de la réponse d'erreur, la réponse énumère les faits qui ont été choisis.

C> Feat
S> 211- Features supported
S> MLST Type*;Size*;Modify*;Perm*;Unique*;UNIX.mode;UNIX.chgd;X.hidden;
S> 211 End
C> Opts MLST
S> 200 MLST OPTS
C> Feat
S> 211- Features supported
S> MLST Type;Size;Modify;Perm;Unique;UNIX.mode;UNIX.chgd;X.hidden;

S> 211 End
C> MLst tmp
S> 250- Listing tmp
S> /tmp
S> 250 End
C> OPTS mlst unique;size;
S> 200 MLST OPTS Size;Unique;
C> MLst tmp
S> 250- Listing tmp
S> Unique=keVO1+YZ5; /tmp
S> 250 End
C> OPTS mlst unique;type;modify;
S> 200 MLST OPTS Type;Modify;Unique;
C> MLst tmp
S> 250- Listing tmp
S> Type=dir;Modify=19990930152225;Unique=keVO1+YZ5; /tmp
S> 250 End
C> OPTS mlst fish;cakes;
S> 200 MLST OPTS
C> MLst tmp
S> 250- Listing tmp
S> /tmp
S> 250 End
C> OptS Mlst Modify;Unique;
S> 200 MLST OPTS Modify;Unique;
C> MLst tmp
S> 250- Listing tmp
S> Modify=19990930152225;Unique=keVO1+YZ5; /tmp
S> 250 End
C> opts MLst fish cakes;
S> 501 Invalid MLST options
C> MLst tmp
S> 250- Listing tmp
S> Modify=19990930152225;Unique=keVO1+YZ5; /tmp
S> 250 End

Cet exemple montre l'effet d'un changement des faits demandés sur les commandes MLST suivantes. Remarquer qu'une erreur de syntaxe laisse inchangé l'ensemble de faits choisi. Remarquer aussi qu'il y a exactement deux espaces précédant le nom de chemin lorsque aucun fait n'est choisi, soit délibérément, soit parce que aucun des faits demandés n'était disponible.

8. Impact sur les autres commandes FTP

Avec l'introduction de MLST, les commandes FTP traditionnelles doivent être étendues pour permettre l'utilisation de plus que les jeux de caractères US-ASCII [ANSI-X3.4] ou EBCDIC. En général, la prise en charge de MLST exige la prise en charge de jeux de caractères arbitraires chaque fois que les noms de fichier et répertoire sont permis. Cela s'applique également aux arguments données dans les commandes suivantes et aux réponses qui en viennent, selon le cas approprié.

APPE	RMD
CWD	RNFR
DELE	RNTO
MKD	STAT
PWD	STOR
RETR	STOU

Les arguments de toutes ces commandes devraient être traités de la même façon que sont traitées les commandes et réponses MLST par rapport au traitement des espaces incorporées, des CR et des NUL. Voir au paragraphe 2.2.

9. Jeux de caractères et internationalisation

Les commandes FTP sont des éléments de protocole, et sont toujours exprimées en ASCII. Les réponses FTP sont composées de code numérique, qui est un élément de protocole, et d'un message, dont on attend souvent qu'il porte des informations pour l'usager. On ne s'attend pas à ce que normalement les usagers interagissent directement avec les éléments de protocole, mais plutôt que le processus d'utilisateur FTP construise les commandes, et interprète les résultats, de la manière qui convient le mieux à cet utilisateur particulier. Le texte explicatif dans les réponses n'a généralement pas une signification particulière pour le protocole. Les codes numériques donnent toutes les informations nécessaires. Les serveurs-PI sont libres de fournir les textes dans tout langage qui peut être adéquatement représenté en ASCII, ou lorsque un autre langage et une autre représentation ont été négociés (voir la [RFC2640]) dans ce langage et représentation.

Les noms de chemins sont supposés être codés en UTF-8, permettant essentiellement que tout caractère soit représenté dans un nom de chemin. Les noms de chemin significatifs sont définis par le serveur NVFS.

Aucune restrictions ne s'impose au contenu des fichiers transférés à l'aide des protocoles FTP. Sauf si le fait "type-de-support" est fourni dans une réponse MLSx, aucun avis n'est donné ici qui pourrait permettre de déterminer le type de contenu. Ces informations sont supposées être obtenues via d'autres moyens.

10. Considérations relatives à l'IANA

La présente spécification utilise des listes de valeurs actuellement tenues par l'IANA, et crée deux nouvelles listes à tenir par l'IANA. Elle n'ajoute aucune nouvelle valeur à un registre existant.

Les registres existants de l'IANA utilisés par la présente spécification sont modifiés en utilisant les mécanismes spécifiés par ailleurs.

10.1 Registre de faits spécifiques du système d'exploitation

Un registre des noms de faits spécifiques d'OS devra être tenu par l'IANA. Les noms des OS pour la portion OS du nom de fait doivent être pris dans la liste de l'IANA des noms d'OS enregistrés. Pour ajouter des noms de fait à ce registre spécifique d'OS des faits spécifiques d'OS, un demandeur doit envoyer à l'IANA une demande, dans laquelle est spécifié le nom de l'OS, le nom du fait spécifique d'OS, une définition de la syntaxe de la valeur du fait, qui doit être conforme à la syntaxe de jeton donnée dans le présent document, et une spécification de la sémantique à associer au fait particulier et à ses valeurs. À réception d'une telle demande, et si la combinaison du nom d'OS et du nom de fait spécifique d'OS n'a pas été précédemment définie, l'IANA ajoutera la spécification au registre.

Tout exemple de fait spécifique d'OS trouvé dans le présent document est à traiter comme un exemple de fait spécifique d'OS possible, et ne fait pas partie du registre IANA simplement parce qu'il serait inclus dans le présent document.

10.2 Registre de type de fichier spécifique du système d'exploitation

Un registre des types de fichier spécifique de l'OS devra être tenu par l'IANA. Les noms d'OS pour la portion OS du nom de fait doivent être tirés de la liste de l'IANA des noms d'OS enregistrés. Pour ajouter un type de fichier à ce registre spécifique des OS des types de fichier spécifique de l'OS, un demandeur doit envoyer à l'IANA une demande, dans laquelle sera spécifié le nom de l'OS, le type de fichier spécifique d'OS, une définition de la syntaxe de la valeur du fait, qui doit être conforme à la syntaxe de jeton donnée dans le présent document, et une spécification de la sémantique à associer au fait particulier et à ses valeurs. À réception d'une telle demande, et si la combinaison du nom d'OS et du type de fichier spécifique d'OS n'a pas été précédemment définie, l'IANA ajoutera la spécification au registre.

Tout exemple de type de fichier spécifique d'OS qui se trouve dans le présent document est à traiter comme un exemple de fait spécifique d'OS possible, et ne fait pas partie du registre IANA simplement parce qu'il serait inclus dans le présent document.

11. Considérations sur la sécurité

Le présent mémoire de concerne pas directement la sécurité. On ne pense pas que les mécanismes documentés ici impactent de quelque façon que ce soit la sécurité de FTP.

La mise en œuvre de la commande SIZE, et peut-être de certains faits des commandes MLSx, peut imposer une charge

considérable au serveur, qui pourrait conduire à des attaques de déni de service. Les serveurs ont cependant mis cela en œuvre depuis de nombreuses années, sans que soient rapportées de difficultés significatives.

Le serveur FTP devrait veiller à ne pas révéler des informations sensibles à des parties non autorisées. En particulier, certains systèmes de fichiers sous-jacents fournissent un identifiant de fichier qui, si il est connu, peut permettre que soient outrepassés de nombreux mécanismes de protection de fichiers systèmes. Cet identifiant ne serait pas un bon choix d'utilisation comme base de la valeur du fait unique.

Les commandes FEAT et OPTS peuvent être produites avant que l'authentification FTP ait été faite [RFC2389]. Cela permet à des clients non authentifiés de déterminer quelles caractéristiques définies ici sont prises en charge, et de négocier la liste de faits pour le résultat MLSx. Aucune commande MLSx ne peut cependant être produite, et on ne prévoit aucun problème découlant de la permission de choisir le format avant l'authentification.

Un exposé général des questions relatives à la sécurité de FTP se trouve dans la [RFC2577].

12. Références normatives

[10646-1] ISO/IEC 10646-1:1993 "Universal multiple-octet coded character set (UCS) -- Part 1: Architecture and basic multilingual plane", International Standard -- Information Technology, 1993.

[ANSI-X3.4] ANSI X3.4-1986 "Coded Character Set--7-bit American Standard Code for Information Interchange".

[IANA] Internet Assigned Numbers Authority. http://www.iana.org ; mél : iana@iana.org

[RFC0854] J. Postel et J. Reynolds,"Spécification du protocole TELNET", STD 8, mai 1983.

[RFC0959] J. Postel et J. Reynolds, "Protocole de transfert de fichiers (FTP)", STD 9, octobre 1985.

[RFC1123] R. Braden, éditeur, "Exigences pour les hôtes Internet – Application et prise en charge", STD 3, octobre 1989.

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

[RFC2389] P. Hethmon, R. Elz, "Mécanisme de négociation de caractéristiques pour FTP", août 1998. *(P.S.)*

[RFC2577] M. Allman, S. Ostermann, "Considérations sur la sécurité de FTP", mai 1999. *(Information)*

[RFC2640] B. Curtin, "Internationalisation du protocole de transfert de fichier", juillet 1999. *(P.S.)*

[RFC3629] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", STD 63, novembre 2003.

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

[RFC4646] A. Phillips, M. Davis, "Étiquettes d'identification des langues", BCP0047 septembre 2006. *(Remplacée par* RFC5646*)*

Remerciements

Le présent document a été produit par le groupe de travail FTPEXT de l'IETF. Les personnes suivantes ont contribué à ce document : Alex Belits, D. J. Bernstein, Dave Cridland, Martin J. Duerst, Bill Fenner (et le reste de l'IESG) Paul Ford-Hutchinson, Mike Gleason, Mark Harris, Stephen Head, Alun Jones, Andrew Main, James Matthews, Luke Mewburn, Jan Mikkelsen, Keith Moore, Buz Owen, Mark Symons, Stephen Tihor, et la totalité du groupe de travail FTPEXT de l'IETF.

Mes excuses pour tous ceux qui auraient été omis par inadvertance.

La description des modifications à la commande REST et aux commandes MDTM et SIZE vient d'un ensemble de modifications suggérées pour le STD 9, la RFC 959 par Rick Adams en 1989. Un document contenant juste ces commandes, édité par David Borman, a été fusionné avec le présent document.

Mike Gleason, Alun Jones et Luke Mewburn ont fourni l'accès aux serveurs FTP utilisés dans certains des exemples.

Tous les exemples du présent document sont tirés d'échanges client/serveur réels, bien que certains aient été coupés par souci de concision, ou pour satisfaire à des contraintes de formatage.

Note de l'éditeur des RFC :
Plusieurs des exemples du présent document excèdent la longueur de ligne standard de 72 caractères des RFC. Comme le présent document est le résultat en, cours de normalisation d'un groupe de travail de l'IETF et qu'il est important pour une sous-communauté de l'IETF, l'éditeur des RFC le publie avec ces violations de marge. Ceci ne constitue pas un précédent.

Adresse de l'auteur

Paul Hethmon
Hethmon Software
10420 Jackson Oaks Way, Suite 201
Knoxville, TN 37922
USA
mél : phethmon@hethmon.com

Déclaration complète de droits de reproduction

Propriété intellectuelle

Remerciement

Le financement de la fonction d'édition des RFC est actuellement fourni par l'Internet Society.