Groupe de travail Réseau
Request for Comment : 2229
Catégorie : Information
Traduction Claude Brière de L'Isle

R. Faith, U. North Carolina, Chapel Hill
B. Martin, Miranda Productions
octobre 1997

Protocole de serveur de dictionnaire

Statut du présent mémoire
Le présent mémoire apporte des informations pour la communauté de l'Internet. Il ne spécifie aucune sorte de norme de l'Internet. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le protocole de serveur de dictionnaire (DICT, *Dictionary Server Protocol*) est un protocole de question/réponse fondé sur des transactions TCP qui permet à un client d'accéder à des définitions de dictionnaire à partir d'un ensemble de bases de données de dictionnaires en langage naturel.

Table des matières

1. Introduction

Pendant de nombreuses années, la communauté de l'Internet s'est appuyée sur le protocole "webster" pour l'accè aux définitions de langage naturel. Le protocole webster prend en charge l'accès à un seul dictionnaire et (facultativement) à un seul thésaurus. Ces dernières années, le nombre de serveurs webster en accès public sur l'Internet a subi une diminution dramatique.

Heureusement, plusieurs dictionnaires et lexiques en accès gratuit sont récemment devenus accessibles sur l'Internet. Cependant, ces bases de données en accès gratuit ne sont pas accessibles via une interface uniforme, et à partir d'un seul site. Ils sont souvent petits et individuellement incomplets, mais pourraient collectivement fournir une base de données intéressante et utile de mots anglais. On a l'exemple du fichier Jargon [JARGON], de la base de données WordNet [WORDNET], de la version de MICRA du dictionnaire Webster non abrégé révisé version 1913 [WEB1913], et le Dictionnaire gratuit en ligne de l'informatique [FOLDOC]. Des dictionnaires de traduction et des dictionnaires non anglais deviennent aussi disponibles (par exemple, le dictionnaire FOLDOC est en cours de traduction en espagnol).

Le protocole webster ne convient pas pour donner accès à un grand nombre de bases de données séparées de dictionnaires, et les extensions au protocole webster actuel ne semblent pas une solution appropriée au problème d'une base de données de dictionnaire.

Le protocole DICT est conçu pour fournir l'accès à plusieurs bases de données. Les définitions de mot peuvent être demandées, l'index des mots peut être parcouru (en utilisant un ensemble facilement extensible d'algorithmes), des informations sur le serveur peuvent être fournies (par exemple, les stratégies de recherche d'index qui sont prises en charge, ou les bases de données qui sont disponibles), et des informations sur une base de données peuvent être fournies (par exemple, informations de copyright, citation, ou distribution). De plus, le protocole DICT a des facilités qui peuvent être utilisées pour interdire l'accès à certaines des bases de données ou à toutes.

1.1 Exigences

Dans ce document, on adopte la convention exposée au paragraphe 1.3.2 de la [RFC1122] d'utiliser les mots en majuscules DOIT, EXIGE, DEVRAIT, RECOMMADE, PEUT et FACULTATIF pour définir la signification de chaque exigence particulière spécifiée dans le document.

En bref: "DOIT" (ou "EXIGE") signifie que l'élément est une exigence absolue de la spécification; "DEVRAIT" (ou "RECOMMANDE") signifie qu'il peut exister des raisons valides pour ignorer cet élément mais que toutes les implications devraient en être comprises avant de ce faire, et "PEUT" (ou "FACULTAIF") signifie que cet élément est facultatif, et peut être omis sans précaution particulière.

2. Survol du protocole

2.1 Niveau Liaison

Le protocole DICT suppose un flux de données fiable tel que celui fourni par TCP. Lorsque TCP est utilisé, un serveur DICT écoute sur l'accès 2628.

Ce serveur est seulement une interface entre les programmes et les bases de données de dictionnaire. Il n'effectue aucune interaction d'utilisateur ou de fonction de niveau présentation.

2.2. Jetons lexicaux

Les commandes et les réponses sont composées de caractères du jeu de caractères UCS [ISO10646] utilisant le codage UTF-8 [RFC2044]. Plus précisément, en utilisant les conventions grammaticales tirées de la [RFC822] :

```
                                        ; (Octal        Décimal)
CHAR    = <tout caractère UTF-8 (1 à 6 octets)>   ; ( 0 - 37,         0 - 31.)
CTL     = <tout caractère de contrôle ASCII et DEL>  ; (  177,127.)
CR      = <ASCII CR, retour charriot>             ; (   15,13.)
LF      = <ASCII LF, fin de ligne>                ; (   12,10.)
SPACE   = <ASCII SP, espace>                      ; (   40,32.)
HTAB    = <ASCII HT, tabulation horizontale>      ; (   11,9.)
```

```
<">         = <ASCII guillemets>                    ; (    42,34.)
<'>         = <ASCII guillemet simple>              ; (    47,          39.)
CRLF        = CR LF
WS          = 1*(SPACE / HTAB)

dqstring    = <"> *(dqtext/quoted-pair) <">
dqtext      = <tout CHAR excepté <">, "\", et les CTL>
sqstring    = <'> *(sqtext/quoted-pair) <'>
sqtext      = <tout CHAR excepté <'>, "\", et les CTL>
quoted-pair = "\" CHAR

atom        = 1*<tout CHAR excepté SPACE, CTL, <'>, <">, et "\">
string      = *<dqstring / sqstring / quoted-pair>
word        = *<atom / string>
description = *<word / WS>
text        = *<word / WS>
```

2.3 Commandes

Les commandes consistent en un mot de commande suivi par zéro, un ou plusieurs paramètres. Les commandes avec paramètres doivent séparer les paramètres les uns des autres et de la commande d'un ou plusieurs espaces ou caractères de tabulation. Les lignes de commande doivent être complètes avec tous les paramètres requis, et il n'est pas permis qu'elles contiennent plus d'une commande.

Chaque ligne de commande doit être terminée par un CRLF.

La grammaire des commandes est :

```
command   = cmd-word *<WS cmd-param>
cmd-word  = atom
cmd-param = database / strategy / word
database  = atom
strategy  = atom
```

Les commandes ne sont pas sensibles à la casse.

Les lignes de commande NE DOIVENT PAS excéder 1 024 caractères en longueur, en comptant tous les caractères y compris les espaces, les séparateurs, la ponctuation, et le CRLF de queue. Il n'y a aucune disposition pour la continuation des lignes de commande. Comme UTF-8 peut coder un caractère en utilisant jusqu'à 6 octets, la mémoire tampon de ligne de commande DOIT être capable d'accepter jusqu'à 6 144 octets.

2.4 Réponses

Les réponses sont de deux sortes, d'état et textuelles.

2.4.1 Réponses d'état

Les réponses d'état indiquent la réponse du serveur à la dernière commande reçue du client.

Les lignes de réponse d'état commencent par un code numérique à trois chiffres qui est suffisant pour distinguer toutes les réponses. Certains d'entre eux peuvent annoncer la transmission de texte qui va suivre.

Le premier chiffre de la réponse indique en gros le succès, l'échec, ou la progression de la commande précédente (généralement sur la base des [RFC0640, RFC0821]) :
1yz – réponse préliminaire positive
2yz – réponse d'achèvement positive
3yz – réponse intermédiaire positive
4yz – réponse d'achèvement transitoire négative
5yz – réponse d'achèvement permanente négative

Le chiffre suivant du code indique la catégorie de la réponse :
x0z – syntaxe
x1z – information (par exemple, aide)
x2z – connexions
x3z – authentification
x4z – encore non spécifié
x5z – système DICT (ces réponses indiquent l'état du système DICT receveur vis-à-vis du transfert ou autre action de système DICT demandée.)
x8z – extensions non standard (mise en œuvre privée)

Les codes de réponse exacts qui devraient être attendus de chaque commande sont précisés dans la description de cette commande.

Certaines réponses d'état contiennent des paramètres tels que des nombres et des chaînes de texte. Le nombre et le type de tels paramètres est fixé pour chaque code de réponse pour simplifier l'interprétation de la réponse. D'autres réponses d'état n'exigent pas d'identifiant de texte spécifique. Les exigences des paramètres sont précisées dans la description des commandes pertientes. Sauf pour les paramètres qui sont détaillés de façon spécifique, le texte qui suit les codes de réponse dépend du serveur.

Les paramètres sont séparés du code de réponse numérique et de chacun des autres par une seule espace. Tous les paramètres numériques sont en décimal, et peuvent avoir des zéros en tête. Toutes les productions de chaînes de paramètres DOIVENT se conformer à la grammaire "atom" ou "dqstring".

Si aucun paramètre n'est présent, et si la mise en œuvre de serveur ne fournit pas de texte spécifique de la mise en œuvre, il PEUT alors y avoir ou pas une espace après le code de réponse.

Les codes de réponse non spécifiés dans la présente norme peuvent être utilisés pour toute commande supplémentaire spécifique de l'installation non elle-même spécifiée.

Celles-ci devraient être choisies de façon à entrer dans le schéma x8z spécifié ci-dessus. L'utilisation de codes de réponse non spécifiés dans une commande standard est interdite.

2.4.2 Réponses d'état générales

En réponse à toute commande, les réponses d'état générales suivantes sont possibles :
500 Erreur de syntaxe, commande non reconnue
501 Erreur de syntaxe, paramètre illégal
502 Commande non mise en œuvre
503 Paramètre de commande non mis en œuvre
420 Serveur temporairement indisponible
421 Serveur fermé à la demande de l'opérateur

2.4.3 Réponses de texte

Avant l'envoi du texte sera envoyée une ligne de réponse d'état numérique, utilisant un code 1yz, qui indique que du texte va suivre. Le texte est envoyé comme une série de lignes successives de sujet textuel, chacune terminée par un CRLF. Une seule ligne contenant seulement un point (code décimal 46, ".") est envoyé pour indiquer la fin du texte (c'est-à-dire que le serveur va envoyer un CRLF à la fin de la dernière ligne de texte, un point, et un autre CRLF).

Si une ligne de texte original contenait un point comme premier caractère de la ligne, ce premier point sera doublé par le serveur DICT. Donc le client doit examiner le premier caractère de chaque ligne reçue. Celles qui commencent par deux points à la suite doivent avoir ces deux points fusionnés en un seul. Celles qui ne contiennent qu'un seul point suivi par un CRLF indiquent la fin de la réponse de texte.

Si la commande OPTION MIME a été donnée, toutes les réponses textuelles seront préfacées par un en-tête MIME [RFC2045] suivi par une seule ligne blanche (CRLF). Voir au paragraphe 3.10.1 des détails sur OPTION MIME.

À la suite d'une réponse de texte, un code de réponse 2yz sera envoyé.

Les lignes de texte NE DOIVENT PAS excéder une longueur de 1 024 caractères, en comptant tous les caractères y compris les espaces, les séparateurs, la ponctuation, le point initial supplémentaire (si nécessaire), et le CRLF de queue. Comme UTF-8 peut coder un caractère en utilisant jusqu'à six octets, la mémoire tampon de ligne de texte DOIT être

capable d'accepter jusqu'à 6 144 octets.

Par défaut, le texte des définitions DOIT être composé de caractères du jeu de caractères UCS [ISO10644] utilisant le codage UTF-8 [RFC2044]. Le codage UTF-8 présente l'avantage de conserver la gamme complète des valeurs de l'US ASCII [USASCII] à 7 bits. Les clients et les serveurs DOIVENT accepter l'UTF-8, même si c'est seulement de façon minimale.

3. Détails des commandes et des réponses

Ci-dessous sont détaillées chaque commande DICT et les réponses appropriées. Chaque commande est présentée en majuscule dans un souci de clarté, mais le serveur DICT est insensible à la casse.

Sauf pour les commandes AUTH et SASLAUTH, chaque commande décrite dans cette section DOIT être mise en œuvre par tous les serveurs DICT.

3.1 Connexion initiale

Lorsqu'un client se connecte initiallement à un serveur DICT, un code 220 est envoyé si l'adresse IP du client est admise en connexion :

220 capacités-de-texte identifiant-de-message

Le code 220 est une bannière, contenant habituellement le nom d'hôte et les informations de version de serveur DICT.

La séquence de caractères du second au dernier dans la bannière est la chaîne des capacités facultatives, qui va permettre aux serveurs de déclarer leur prise en charge des extensions au protocole DICT. La chaîne des capacités est définie ci-dessous :

```
capacités    = ["<" msg-atom *("." msg-atom) ">"]
msg-atom     = 1*<tout CHAR excepté SPACE, CTL, "<", ">", ".", et "\">
```

Les capacités individuelles sont décrites par un seul msg-atom. Par exemple, la chaîne <html.gzip> pourrait être utilisée pour décrire un serveur qui prend en charge des extensions qui permettent une sortie HTML ou compressée. Le noms des capacités qui commencent par "x" ou "X" sont réservées pour des extensions experimentales, et NE DEVRAIENT PAS être défines dans une future spécification de protocole DICT. Certaines de ces capacités peuvent informer le client que certaine fonctionnalité est disponible ou peut être demandée. Les capacités suivantes sont actuellement définies :

mime	La commande OPTION MIME est prise en charge
auth	La commande AUTH est prise en charge
kerberos_v4	Le mécanisme SASL Kerberos version 4 est pris en charge
gssapi	Le mécanisme SASL GSSAPI [RFC2078] est pris en charge
skey	Le mécanisme SASL S/Key [RFC1760] est pris en charge
external	Le mécanisme SASL externe est pris en charge

La dernière séquence de caractères dans la bannière est un identifiant de message msg-id, similaire au format spécifié dans la [RFC822]. La description simplifiée est donnée ci-dessous

```
msg-id       = "<" spec ">"                  ; identifiant de message unique
spec         = partie-locale "@" domaine
partie-locale = msg-atom *("." msg-atom)
domaine      = msg-atom *("." msg-atom)
```

Noter que, à l'opposé de la [RFC822], les espaces et paires entre guillemets ne sont pas admis dans le msg-id. Cette restriction rend le msg-id plus aisé à localiser et analyser pour le client mais ne diminue pas de façon significative la sécurité car la longueur de l'identifiant de message est arbitraire (dans les limites de la longueur de réponse établie dans une autre section de ce document).

Noter aussi que les crochets ouvert et fermé font partie de l'identifiant de message et devraient être inclus dans la chaîne qui est utilisée pour le calcul de la somme de contrôle MD5.

Cet identifiant de message sera utilisé par le client lors de la formulation de la chaîne d'authentification utilisée dans la commande AUTH.

Si d'adresse IP du client n'est pas admise à la connexion, un code 530 est alors envoyé à la place :

530 Accès refusé

Des réponses de défaillance provisoire sont aussi possibles :

420 Serveur temporairement indisponible
421 Serveur en cours de clôture sur demande de l'opérateur

Par exemple, le code de réponse 420 devrait être utilisé si le serveur ne peut actuellement fourcher un processus de serveur (ou ne peut actuellement obtenir une autre ressources requise pour poursuivre une connexion utilisable), mais s'attend à être capable de faire ce fourchement ou obtenir ces ressources dans un futur proche.

Le code de réponse 421 devrait être utilisé lorsque le serveur a été fermé à la demande de l'opérateur, ou lorsque les conditions indiquent que la capacité à servir plus de demandes dans le futur proche sera impossible. Cela peut être utilisé pour permettre une fermeture temporaire en douceur à la demande de l'opérateur d'un serveur, ou pour indiquer qu'un serveur bien connu a été retiré de façon permanente du service (auquel cas, le message de texte pourra fournir plus d'informations).

3.2 Commande DEFINE

DEFINE database word

3.2.1 Description

Cette commande va chercher le mot spécifié dans la base de données spécifiée. Tous les serveurs DICT DOIVENT mettre en œuvre cette commande.

Si le nom de la base de données est spécifié avec un point d'exclamation (code décimal 33, "!") toutes les bases de données seront alors parcourues jusqu'à ce qu'une correspondance soit trouvée, et toutes les correspondances dans cette base de données seront affichées. Si le nom de la base de données est spécifié avec une étoile (code décimal 42, "*") alors toutes les correspondances dans les bases de données disponibles seront affichées. Dans ces deux cas particuliers, les bases de données seront parcourues dans le même ordre que celui imprimé par la commande "SHOW DB".

Si le mot n'est pas trouvé, le code d'état 552 est alors envoyé.

Si le mot a été trouvé, le code d'état 150 est alors envoyé, ce qui indique qu'une ou plusieurs définitions suivent.

Pour chaque définition, le code d'état 151 est envoyé, suivi par le corps textuel de la définition. Les trois premiers paramètres délimités par une espace qui suivent le code d'état 151 donnent le mot restitué, le nom de la base de données (qui est le même que la première colonne de la commande SHOW DB), et une courte description pour la base de données (qui est la même que la seconde colonne de la commande SHOW DB). Le nom court est convenable pour l'impression comme :

From name:

avant l'impression de la définition. Cela fournit l'information de source pour l'utilisateur.

Le corps textuel de chaque définition est terminé par une séquence CRLF point CRLF.

Après que toutes les définitions ont été envoyées, le code d'état 250 est envoyé. Cette commande peut fournir des informations facultatives de temps (qui dépendent du serveur et ne sont pas destinées à être analysables par le client). Ces informations supplémentaires sont utiles lors du débogage et du réglage du serveur.

3.2.2 Réponses

550 Base de données invalide, utiliser "SHOW DB" pour la liste des bases de données
552 Pas de correspondance

150 n définitions restituées – les définitions suivent
151 nom de la base de données de mots – le texte suit
250 ok (information facultatives de temps fournies)

Les codes de réponse 150 et 151 exigent des paramètres particuliers au titre du texte. Le client peut utiliser ces paramètres pour afficher les informations sur le terminal de l'utilisateur.

Pour le code 150, paramètre 1 indique le nombre de définitions restituées.

Pour le code 151, paramètre 1 est le mot restitué, paramètre 2 est le nom de la base de données (le premier nom est celui indiqué par "SHOW DB") à partir duquel la définition a été restituée, et paramètre 3 est la description courte de la base de données (la seconde colonne de la commande "SHOW DB").

3.3 Commande MATCH

MATCH database strategy word

3.3.1 Description

Cette commande recherche un indice pour le dictionnaire, et rapporte les mots qui ont été trouvés en utilisant une stratégie particulière. Toutes les stratégies ne sont pas utiles pour tous les dictionnaires, et certains dictionnaires peuvent accepter des stratégies de recherche supplémentaires (par exemple, recherche inverse). Tous les serveurs DICT DOIVENT mettre en œuvre la commande MATCH, et DOIVENT prendre en charge les stratégies "exact" et "prefix". Elles sont faciles à mettre en œuvre et sont généralement les plus utiles. Les autres stratégies dépendent du serveur.

La stratégie "exact" correspond exactement à un mot, bien que différents serveurs puissent traites différemment des données non alphanumériques. Nous avons trouvé qu'une comparaison insensible à la casse qui ignore les caractères non alphanumériques et qui replie les espace est utile pour les dictionnaires de langue anglaise. D'autres comparaisons peuvent être plus appropriées pour d'autres langages ou lors de l'utilisation de jeux de caractères étendus.

La stratégie "prefix" est similaire à celle de "exact", sauf qu'elle ne compare que la première partie du mot.

Différents serveurs peuvent mettre en œuvre différemment ces algorithmes. La seule exigence est que ces stratégies qui portent les noms "exact" et "prefix" existent de telle sorte qu'un simple client puisse les utiliser.

Les autres stratégies qui pourraient être envisagées par le développeur d'un serveur sont des correspondances fondées sur des algorithmes de sous-chaîne, de suffixe, d'expressions régulières, de soundex [KNUTH73], et de Levenshtein [PZ85]. Ces deux derniers sont particulièrement utiles pour corriger des erreurs d'orthographe. D'autres stratégies utiles effectuent des sortes de recherche "inverse" (c'est-à-dire, en cherchant des définitions pour trouver le mot suggéré par la question).

Si le nom de la base de données est spécifié avec un point d'exclamation (code décimal 33, "!") toutes les bases de données seront alors parcourues jusqu'à ce qu'une correspondance soit trouvée, et toutes les correspondances dans cette base de données seront affichées. Si le nom de la base de données est spécifié avec une étoile (code décimal 42, "*"), alors toutes les correspondances dans toutes les bases de données disponibles seront affichées. Dans ces deux cas particuliers, les bases de données seront parcourues dans le même ordre que celui imprimé par la commande "SHOW DB".

Si la stratégie est spécifiée en utilisant un point (code décimal 46, ".") la correspondance du mot sera alors recherchée en utilisant une stratégie par défaut dépendant du serveur, qui devrait être la meilleure stratégie disponible pour les vérifications d'orthographe interactives. Elles sont habituellement dérivées de l'algorithme de Levenshtein [PZ85].

Si aucune correspondance n'est trouvée dans aucune des bases de données accédées, le code d'état 552 sera retourné.

Autrement, le code d'état 152 sera retourné, suivi par une liste des mots qui correspondent, un par ligne, sous la forme :

database word

Cela rend les réponses directement utiles dans une commande DEFINE.

Le corps textuel de la liste de correspondances est terminé par une séquence CRLF point CRLF.

À la suite de la liste, le code d'état 250 est envoyé, qui peut inclure des informations de temps et de statistiques spécifiques

du serveur, comme exposé dans la section sur la commande DEFINE.

3.3.2 Réponses

550 base de données invalide, utiliser "SHOW DB" pour la liste des bases de données
551 stratégie invalide, utiliser "SHOW STRAT" pour la liste des stratégies
552 pas de dorrespondance
152 n correspondances trouvées – le texte suit
250 ok (plus des informations facultatives d'heure)

Le code de réponse 152 exige un paramètre particulier au titre de son texte. Paramètre 1 doit être le nombre de correspondances restituées.

3.4 Note sur les bases de données virtuelles

La capacité à chercher dans toutes les bases de données en utilisant une seule commande est obtenue en utilisant les bases de données "*" et "!" particulières.

Cependant, parfois, un client peut vouloir chercher sur certaines mais pas sur toutes les bases de données d'un serveur particulier. Une solution de remplacement pour ce client est d'utiliser la commande SHOW DB pour obtenir une liste des bases de données et des descriptions, puis ensuite (peut-être avec une aide humaine), de choisir un sous-ensemble de ces bases de données pour une recherche interactive. Une fois que ce choix a été fait, les résultats peuvent être sauvegardés, par exemple, dans un fichier de configuration client.

Une autre solution de remplacement est que le serveur fournisse des bases de données "virtuelles" qui vont fusionner en une seule plusieurs des bases de données régulières. Par exemple, une base de données virtuelle peut être fournie qui comportera tous les dictionnaires de traduction, mais ne comportera pas les dictionnaires ou encyclopédies ordinaires. Les bases de données "*" et "!" spéciales peuvent être considérées comme des nom de bases de données virtuelles qui donnent accès à toutes les bases de données. Si un serveur met en œuvre des bases de données virtuelles, les bases de données "*" et "!" spéciales devraient probablement exclure d'autres bases de données virtuelles (car elles fournissent simplement des informations dupliquées dans d'autres bases de données). Si des bases de données virtuelles sont prises en charge, elles devraient être listées comme bases de données régulières avec la commande SHOW DB (bien que, comme "*" et "!" sont exigés, il ne soit pas nécessaire de les mettre sur la liste).

Les bases de données virtuelles sont un détail spécifique de la mise en œuvre qui n'a absolument aucun impact sur le protocole DICT. Le protocole DICT voit les bases de donénes virtuelles et non virtuelles de la même façon.

Nous avons cependant mentionné les bases de données virtuelles à cet endroit parce qu'elles résolvent un problème de choix de base de données qui aurait aussi pu être résolu par des changements du protocole. Par exemple, chaque dictionnaire pourrait se voir allouer des attributs, et le protocole pourrait être étendu pour spécifier des recherches sur des bases de données avec certains attributs. Cependant, cela compliquerait sans nécessité l'analyse qui doit être effectuée par la mise en œuvre. De plus, sauf si le système de classification est extrêmement général, il y a le risque que cela restreigne les types de bases de données qui peuvent être utilisées avec le protocole DICT (bien que le protocole ait été conçu en visant les bases de données de langages humains, il est applicable à toute application de base de données en lecture seule, en particulier celles avec une seule clé alphanumérique semi-unique et des données textuelles).

3.5 Commande SHOW

3.5.1 SHOW DB

SHOW DB
SHOW DATABASES

3.5.1.1 Description

Affiche la liste des bases de données actuellement accessibles, une par ligne, sous la forme :

database description

Le corps textuel de la liste de bases de données est terminée par une séquence CRLF point CRLF. Tous les serveurs DICT DOIVENT mettre en œuvre cette commande.

Noter que certaines bases de données peuvent être interdites à cause du domaine du client ou d'un manque d'authentification de l'utilisateur (voir les commandes AUTH et SASLAUTH aux paragraphes 3.11 et 3.12). Des informations sur ces bases de données ne sont disponible qu'après que l'authentification est effectuée. Jusqu'à ce moment, le client va interagir avec le serveur comme si les bases de données supplémentaires n'existaient pas.

3.5.1.2 Réponses

110 n bases de données présentes – le texte suit
554 Aucune base de données n'est présente

Le code de réponse 110 exige un paramètre particulier. Paramètre 1 doit être le nombre de bases de données disponibles pour l'utilisateur.

3.5.2 SHOW STRAT

SHOW STRAT
SHOW STRATEGIES

3.5.2.1 Description

Affiche la liste des stratégies de recherche actuellement prises en charge, une par ligne, sous la forme :

strategy description

Le corps textuel de la liste des stratégies est terminé par une séquence CRLF point CRLF. Tous les serveurs DICT DOIVENT mettre en œuvre cette commande.

3.5.2.2 Réponses

111 n stratégies disponibles – le texte suit
555 pas de stratégie disponible

Le code de réponse 111 exige un paramètre spécial. Paramètre 1 doit être le nombre de stratégies disponnibles.

3.5.3 SHOW INFO

SHOW INFO database

3.5.3.1 Description

Affiche les informations de source, de copyright, et de licence sur la base de données spécifiée. Les informations sont un texte de forme libre et conviennent à l'affichage pour l'utilisateur de la même manière qu'une définition. Le corps du texte des informations est terminé par une séquence CRLF point CRLF. Tous les serveurs DICT DOIVENT mettre en œuvre cette commande.

3.5.3.2 Réponses

550 Base de données invalide, utiliser "SHOW DB" pour la liste des bases de données
112 Les informations de base de données suivent

Ces codes de réponse n'exigent aucun paramètre particulier.

3.5.4 SHOW SERVER

SHOW SERVER

3.5.4.1 Description

Affiche les informations de serveur local écrites par l'administrateur local. Cela peut inclure des informations sur les bases de données ou les stratégies locales, ou des informations administratives telles que qui contacter pour l'accès aux bases de données qui exigent une authentification. Tous les serveurs DICT DOIVENT mettre en œuvre cette commande.

3.5.4.2 Réponses

114 Les informations sur le serveur suivent

Ce code de réponse n'exige aucun paramètre particulier.

3.6 Commande CLIENT

CLIENT text

3.6.1 Description

Cette commande permet au client de fournir des informations sur lui-même pour d'éventuels besoins de connexion et de statistiques. Tous les clients DEVRAIENT envoyer cette commande après connexion au serveur. Tous les serveurs DICT DOIVENT mettre en œuvre cette commande (noter cependant que le serveur n'a rien à faire avec les informations fournies par le client).

3.6.2 Réponses

250 ok (des informations facultatives d'heure prennent place ici)

Ce code de réponse n'exige aucun paramètre particulier.

3.7 Commande STATUS

STATUS

3.7.1 Description

Affiche des informations spécifiques du serveur d'horaire ou de débogage. Ces informations peuvent être utiles pour le débogage ou le réglage d'un serveur DICT. Tous les serveurs DICT DOIVENT mettre en œuvre cette commande (noter cependant que la partie texte de la réponse n'est pas spécifiée et peut être omise).

3.7.2 Réponses

210 (des informations facultatives d'heure et de statistiques prennent place ici)

Ce code de réponse n'exige aucun paramètre particulier.

3.8 Commande HELP

HELP

3.8.1 Description

Donne un bref résumé des commandes qui sont comprises de cette mise en œuvre de serveur DICT. Le texte d'aide sera présenté comme une réponse textuelle, terminée par un seul point sur une ligne spécifique. Tous les serveurs DICT DOIVENT mettre en œuvre cette commande.

3.8.2 Réponses

113 Le texte d'aide suit

Ce code de réponse n'exige aucun paramètre particulier.

3.9 Commande QUIT

QUIT

3.9.1 Description

Cette commande est utilisée par le client pour sortir proprement du serveur. Tous les serveurs DICT DOIVENT mettre en œuvre cette commande.

3.9.2 Réponses

221 Fermeture de la connexion *(Closing Connection)*

Ce code de réponse n'exige aucun paramètre particulier.

3.10 Commande OPTION

3.10.1 OPTION MIME

OPTION MIME

3.10.1.1 Description

Demande que toutes les réponses de texte soient préfacées par un en-tête MIME [RFC2045] suivi par une seule ligne blanche (CRLF).

Si un client demande cette option, le client DOIT être capable d'analyser les en-têtes Type de contenu et Codage de transfert de contenu, et DOIT être capable d'ignorer les réponses textuelles qui ont un contenu ou codage non pris en charge. Un client DOIT accepter le codage UTF-8 [RFC2044], ce qui signifie au minimum que le client DOIT reconnaître les codages UTF-8 multi-octet et les convertir en des symboles qui puissent être imprimés par le client.

Si un client demande cette option, le serveur fournira alors un en-tête MIME. Si l'en-tête est vide, la réponse de texte commencera par une seule ligne blanche (CRLF), auquel cas un client DOIT interpréter cela comme un en-tête par défaut. L'en-tête par défaut pour l'authentification SASL est :

Type de contenu : application/octet-stream
Codage de transfert de contenu : base64

L'en-tête par défaut pour toutes les autres réponses de texte est :

Type de contenu : text/plain; charset=utf-8
Codage de transfert de contenu : 8bit

Si OPTION MIME n'est pas spécifié par le client, le serveur peut alors restreindre le contenu des informations fournies au client. Par exemple, une définition peut être accompagnée d'une image et d'un clip audio, mais le serveur ne peut pas transmettre ces informations à moins que le client ne soit capable d'analyser les en-têtes de format MIME.

Noter que, à cause des restrictions de longueur de ligne et de la sémantique de fin de réponse, le codage de transfert de contenu "binaire" NE DOIT PAS être utilisé. À l'avenir, des extensions au protocole pourraient être fournies pour permettre à un client de demander des codages binaires, mais la valeur par défaut DEVRAIT toujours être que le client peut chercher une séquence "CRLF . CRLF" pour localiser la fin de la réponse de texte en cours. Cela permet aux clients de sauter facilement les réponses de texte qui ont des types ou des codages non pris en charge.

À l'avenir, après une expérience significative de grandes bases de données dans divers langages, et après une évaluation des besoins de spécification de jeux de caractères et d'autres codages (par exemple, de codages compressés ou en BASE64), des extensions standard à ce protocole devraient être proposées pour permettre au client de demander certains types de contenu ou de codages. Il faudra veiller à ce que ces extensions n'exigent pas de prise de contact qui mette en échec le traitement en parallèle. En attendant, des extensions privées devraient être utilisées pour explorer l'espace des paramètres pour déterminer la meilleure mise en œuvre de ces extensions.

OPTION MIME est une capacité de serveur EXIGÉE, tous les serveurs DICT DOIVENT mettre en œuvre cette commande.

3.10.1.2 Réponses

250 ok (les informations facultaives d'heure prennent place ici)

Noter que certaines mises en œuvre de serveur plus anciennes, achevées avant la finalisation du présent document, vont

retourner un code d'état 500 si cette commande n'est pas mise en œuvre. Les clients DEVRAIENT être capables d'accepter ce comportement, en faisant des hypothèses par défaut. Les clients peuvent aussi examiner les chaînes de capacités dans l'en-tête de code d'état 220 pour déterminer si un serveur accepte cette capacité.

3.11 Commande AUTH

AUTH username authentication-string

3.11.1 Description

Le client peut s'authentifier auprès du serveur en utilisant un nom d'utilisateur et un mot de passe. La chaîne d'authentification sera calculée comme dans le protocole APOP exposé dans la [RFC1939]. En bref, la chaîne d'authentification est la somme de contrôle MD5 de l'enchaînement de l'identifiant de message (obtenu de la bannière initiale) et du "secret partagé" qui est mémorisé dans les fichiers de configuration du serveur et du client. Comme l'utilisateur n'a pas à frapper de secret partagé lorsque il accède au serveur, le secret partagé peut être une phrase de passe de longueur arbitraire. À cause de la facilité de calcul de la somme de contrôle MD5, le secret partagé devrait être significativement plus long qu'un mot de passe habituel.

L'authentification peut rendre plus de bases de données de dictionnaire disponibles pour la session en cours. Par exemple, il peut y avoir des bases de données en accès réparti public disponibles à tous les utilisateurs, et d'autres bases de données privées disponibles seulement aux utilisateurs authentifiés. Ou, un serveur peut exiger l'authentification de tous les usagers pour minimiser l'utilisation des ressources sur la machine du serveur.

L'authentification est une capacité de serveur facultative. La commande AUTH PEUT être mise en œuvre par un serveur DICT.

3.11.2 Réponses

230 Authentication réussie
531 Accès refusé, utiliser "SHOW INFO" pour les informations sur le serveur

Ce code de réponse n'exige aucun paramètre particulier.

3.12 Commande SASLAUTH

SASLAUTH mechanism initial-response
SASLRESP response

3.12.1 Description

L'authentification simple avec couche de sécurité (SASL, *Simple Authentication and Security Layer*) est en cours de développement [RFC2222]. Le protocole DICT réserve les commandes SASLAUTH et SASLRESP pour cette méthode d'authentification. Le résultat d'une authentification réussie avec SALSAUTH sera le même que celui de l'authentification AUTH réussie : plus de bases de données de dictionnaires peuvent devenir accessibles lors de la session en cours.

La réponse initiale est un paramètre facultatif pour la commande SASLAUTH, codée en utilisant le codage BASE64 [RFC2045]. Certains mécanismes de SASL peuvent permettre l'utilisation de ce paramètre. Si l'authentification SASL est prise en charge par un serveur DICT, ce paramètre DOIT alors être aussi accepté.

Une authentification SASL normale sera initialisée par le client en utilisant la commande SASLAUTH. Le serveur va répondre avec le code d'état 130, suivi par une mise en cause. La mise en cause sera suivie par le code d'état 330, qui indique que le client doit maintenant envoyer une réponse au serveur.

Selon les détails du mécanisme SASL actuellement utilisé, le serveur va soit continuer l'échange en utilisant le code d'état 130, une mise en cause, et le code d'état 330, soit le serveur va utiliser le code d'état 230 ou 531 pour indiquer que l'authentification a réussi ou échoué.

Les mises en causes envoyées par le serveur sont définies par les mécanismes comme des jetons binaires de longueur arbitraire, et devraient être envoyés en utilisant une réponse de texte DICT standard, comme décrit au paragraphe 2.4.3. Si

OPTION MIME n'est pas mis, le codage BASE64 DOIT alors être utilisé. Si OPTION MIME est mis, BASE64 est alors le codage par défaut, comme spécifié au paragraphe 3.10.1.

Le client enverra toutes les réponses en utilisant la commande SASLRESP et un paramètre codé en BASE64. Les réponses envoyées par le client sont définies par les mécanismes comme des jetons binaires de longueur arbitraire. Se rappeler que les lignes de commande DICT ne peuvent avoir que 1 024 caractères de long, de sorte que les réponse fournies par un client DICT sont limitées.

Si le mécanisme spécifié dans la commande SASLAUTH n'est pas accepté, le code d'état 532 sera retourné.

L'authentification est une capacité de serveur facultative. La commande SASLAUTH PEUT être mise en œuvre par un serveur DICT.

3.12.2 Réponses

130 Mise en cause suit
330 Envoi de réponse
230 Authentification réussie
531 Accès refusé, utiliser "SHOW INFO" pour les informations sur le serveur
532 Accès refusé, mécanisme inconnu

Ce code de réponse n'exige aucun paramètre particulier.

4. Traitement de commandes en parallèle

Tous les serveurs DICT DOIVENT être capables d'accepter des commandes multiples dans une seule opération d'envoi TCP. Utiliser une seule opération d'envoi TCP pour plusieurs commandes peut améliorer significativement les performances de DICT, en particulier en présence de liaisons réseau à forte latence.

Les problèmes de mise en œuvre possibles pour un serveur DICT qui vont empêcher le traitement en parallèle des commandes sont similaires aux problèmes qui empêchent le traitement en parallèle dans un serveur SMTP. Ces problèmes sont exposés en détail dans la [RFC1854], qui devrait être consultée par tous les développeurs de serveur DICT.

La principale implication est qu'une mise en œuvre de serveur DICT NE DOIT PAS purger ou perdre d'une autre façon les contenus de mémoire tampon d'entrée TCP dans quelque circonstance que ce soit.

Un client DICT peut traiter en parallèle plusieurs commandes et doit vérifier les réponses à chaque commande individuelle. Si le serveur a fermé, il est possible que toutes les commandes ne soient pas traitées. Par exemple, un simple client DICT peut traiter en parallèle une séquence de commandes CLIENT, DEFINE, et QUIT lorsque il se connecte au serveur. Si le serveur est fermé, le code de réponse initial envoyé par le serveur peut être 420 (temporairement indisponible) au lieu de 220 (bannière). Dans ce cas, la définition ne peut pas être restituée, et le client devrait rapporter une erreur ou réessayer la commande. Si le serveur fonctionne, il peut être capable de renvoyer la bannière, la définition, et le message de terminaison dans une seule opération d'envoi TCP.

5. Spécification des URL

Le schéma d'URL de DICT est utilisé pour se référer aux définitions ou listes de mots disponibles en utilisant le protocole DICT :

dict://<user>;<auth>@<host>:<port>/d:<word>:<database>:<n>
dict://<user>;<auth>@<host>:<port>/m:<word>:<database>:<strat>:<n>

La syntaxe "/d" spécifie la commande DEFINE (paragraphe 3.2), tandis que "/m" spécifie la commande MATCH (paragraphe 3.3).

Certains des champs "<user>;<auth>@", ":<port>", "<database>", "<strat>", et "<n>" ou tous peuvent être omis.

"<n>" sera habituellement omis, mais lorsqu'il est inclus, il spécifie la n^{ème} définition ou correspondance d'un mot. Une méthode pour extraire exactement cette information du serveur n'est pas disponible en utilisant le protocole DICT.

Cependant, un client qui utilise la spécification d'URL pourrait obtenir toutes les définitions ou correspondances, puis choisir ensuite celle qui est spécifiée.

Si "<user>;<auth>@" est omis, aucune authentification n'est faite. Si ":<port>" est omis, l'accès par défaut (2628) DEVRAIT être utilisé. Si "<database>" est omis, "!" DEVRAIT être utilisé (voir au paragraphe 3.2). Si "<strat>" est omis, "." DEVRAIT être utilisé (voir au paragraphe 3.3).

"<user>;<auth>@" spécifie le nom d'utilisateur et le type d'authentification effectuée. Pour "<auth>", la chaîne "AUTH" indique que l'authentification APOP utilisant la commande AUTH sera effectuée, tandis que la chaîne "SASLAUTH=<auth_type>" indique que les commandes SASLAUTH et SASLRESP seront utilisées, avec "<auth_type>" qui indique le type d'authentification SASL qui sera utilisé. Si "<auth_type>" est une étoile (code décimal 42, "*"), le client choisira alors un type d'authentification.

Chaque fois que l'authentification est exigée, le client DEVRAIT demander des informations supplémentaires (par exemple, une phrase de passe) à l'utilisateur. À l'opposé de la [RFC1738], les mots de passe en clair ne sont pas permis dans l'URL.

Les deux points en queue peuvent être omis. Par exemple, les URL suivants peuvent spécifier des définitions ou correspondances :

```
dict://dict.org/d:shortcake:
dict://dict.org/d:shortcake:*
dict://dict.org/d:shortcake:wordnet:
dict://dict.org/d:shortcake:wordnet:1
dict://dict.org/d:abcdefgh
dict://dict.org/d:sun
dict://dict.org/d:sun::1
dict://dict.org/m:sun
dict://dict.org/m:sun::soundex
dict://dict.org/m:sun:wordnet::1
dict://dict.org/m:sun::soundex:1
dict://dict.org/m:sun:::
```

6. Extensions

Ce protocole a été conçu de façon à ce que les bases de données de texte plat puissent être utilisées avec un serveur après un minimum d'analyse et de formatage. D'après notre expérience, il est suffisant de simplement construire un index pour une base de données pour qu'elle soit utilisable avec un serveur DICT. La capacité à servir un texte préformaté est particulièrement importante dans la mesure où des bases de données en accès libre sont souvent distribuées comme fichiers de texte plat sans aucune information de balisage sémantique (et contiennent souvent du "ASCII art" qui empêche l'automatisation d'un formatage même simple).

Cependant, si on a une base de données avec des informations de balisage suffisantes, il peut être possible de générer en sortie divers formats différents (par exemple, du simple HTML ou du SGML plus sophistiqué). La spécification du formatage sort du domaine d'application de ce document. Les exigences de négociation de format (y compris de jeu de caaractères et d'autres codages) sont complexes et devraient être examinées au fil du temps avec un peu plus d'expérience. Nous suggérons que l'utilisation de formats différents, ainsi que d'autres caractéristiques de serveur, soient explorées au titre des extensions au protocole.

6.1 Syntaxe des commandes expérimentales

Les commandes d'une seule lettre sont réservées au débogage et aux essais, et NE DEVRAIENT PAS être définies dans une future spécification de protocole DICT, et NE DOIVENT PAS être utilisées par un logiciel client.

Les commandes qui commencent par la lettre "X" sont réservées aux extensions expérimentales, et NE DEVRAIENT PAS être définies dans de futures spécifications du protocole DICT. Les auteurs de logiciels clients devraient comprendre que ces commandes ne font pas partie du protocole DICT et peuvent n'être pas disponibles sur tous les serveurs DICT.

6.2 Commandes expérimentales et traitement en parallèle

Les commandes expérimentales devraient être conçues de telle sorte qu'un client puisse traiter en parallèle les commandes expérimentales sans savoir si un serveur prend en charge cette commande (par exemple, au lieu d'utiliser la négociation des caractéristiques). Si le serveur ne prend pas en charge la commande, un code de réponse dans la série 5yz (généralement 500) sera alors donné, notifiant au client que l'extension n'est pas prise en charge. Bien sûr, selon la complexité des extensions ajoutées, la négociation des caractéristiques peut être nécessaire. Pour aider à minimiser le temps de négociation, les caractéristiques prises en charge par les serveurs peuvent être annoncées dans la bannière (code 220) en utilisant le paramètre de capacités facultatives.

7. Sommaire des codes de réponse

Ci-dessous figure une récapitulation des codes de réponse. Une étoile (*) dans la première colonne indique que la réponse a des arguments définis qui doivent être fournis.

* 110 n bases de données présentes – texte suit
* 111 n stratégies disponibles – texte suit
  112 des informations de base de données suivent
  113 du texte d'aide suit
  114 des informations de serveur suivent
  130 une mise en cause suit
* 150 n définitions restituées – les définitions suivent
* 151 nom de base de données de mots – texte suit
* 152 n correspondances trouvées – texte suit
  210 (des informations facultatives d'heure et statistiques prennent place ici)
* 220 identifiant de message de texte
  221 clôture de connexion
  230 authentification réussie
  250 ok (des informations facultatives d'heure prennent place ici)
  330 envoi de réponse
  420 serveur temporairement indisponible
  421 serveur fermant à la demande de l'opérateur
  500 erreur de syntaxe, commande non reconnue
  501 erreur de syntaxe, paramètres illégaux
  502 commande non mise en œuvre
  503 paramètre de commande non mis en œuvre
  530 accès refusé
  531 accès refusé, utiliser "SHOW INFO" pour des informations de serveur
  532 accès refusé, mécanisme inconnu
  550 base de données invalid, utiliser "SHOW DB" pour la liste des bases de données
  551 stratégie invalide, utiliser "SHOW STRAT" pour une liste des stratégies
  552 pas de correspondance
  554 aucune base de données n'est présente
  555 pas de stratégie disponible

8. Exemples de conversations

Voici des exemples de conversations qu'on pourrait s'attendre à avoir avec un serveur DICT normal. La notation "C:" indique les commandes établies par le client, et "S:" indique les réponses envoyées par le serveur. Les lignes blanches sont incluses pour la clarté de la présentation et n'indiquent pas de nouvelles lignes réelles dans la transaction.
(NDT : le texte entre crochets [xxx] *est celui des commentaires sur les commandes, le texte en italique entre parenthèses est celui de la traduction des commandes.)*

8.1 Exemple 1 - commandes HELP, DEFINE et QUIT

C: [le client initialise la connexion]
S: 220 dict.org dictd (version 0.9) <27831.860032493@dict.org>
C: HELP *(aide)*
S: 113 Help text follows *(un texte d'aide suit)*

S: DEFINE database word look up word in database *(définir un mot de la base de données, chercher un mot dans la base)*
S: MATCH database strategy word match word in database using strategy *(faire correspondre un mot de la base de données en utilisant une stratégie)*
S: [plus de texte d'aide selon le serveur]
S: .
S: 250 Command complete *(commande terminée)*

C: DEFINE ! penguin *(définir pingouin)*

S: 150 1 definitions found: list follows *(1 définition trouvée : texte suit)*
S: 151 "penguin" wn "WordNet 1.5" : definition text follows *("pingouin" dans WordNet 1.5 : le texte de la définition suit)*
S: penguin
S: 1. n: short-legged flightless birds of cold southern esp. Antarctic *(oiseau non volant à pattes courtes des régions sud)*
S: regions having webbed feet and wings modified as flippers *(froides, en particulier de l'Antartique, à pieds palmés)*
S: . *(et aux ailes modifiées en nageoires)*
S: 250 Command complete *(commande terminée)*

C: DEFINE * shortcake *(définir * sablé)*

S: 150 2 definitions found: list follows *(2 définitions trouvées : la liste suit)*
S: 151 "shortcake" wn "WordNet 1.5" : text follows *("sablé" dans WordNet 1.5 : le texte de la définition suit)*
S: shortcake *(sablé)*
S: 1. n: very short biscuit spread with sweetened fruit and usu. *(1. n : très petit biscuit tartiné avec de la confiture de fruit)*
S: whipped cream *(et habituellement de la crème fouettée)*
S: .
S: 151 "Shortcake" web1913 "Webster's Dictionary (1913)" : text follows
S: Shortcake
S: \Short"cake`\, n.
S: An unsweetened breakfast cake shortened with butter or lard, *(gateau de petit déjeuner non sucré à pâte beurrée)*
S: rolled thin, and baked. *(ou lardée, roulé finet cuît au four.)*
S: .
S: 250 Command complete *(commande terminée)*

C: DEFINE abcdefgh *(définir abcdefgh)*
S: 552 No match *(pas de correspondance)*

C: quit *(quitter)*
S: 221 Closing connection *(clôture de la connexion)*

8.2 Exemple 2 – commande SHOW, commande MATCH

C: SHOW DB *(montrer les bases de données)*

S: 110 3 databases present: list follows *(3 bases de données présentes, la liste suit)*
S: wn "WordNet 1.5"
S: foldoc "Free On-Line Dictionary of Computing"
S: jargon "Hacker Jargon File"
S: .
S: 250 Command complete

C: SHOW STRAT *(montrer les stratégies)*

S: 111 5 strategies present: list follows *(5 stratégies présentes : la liste suit)*
S: exact "Match words exactly" *(correspondant exactement au mot)*
S: prefix "Match word prefixes" *(correspondant au préfixe du mot)*
S: substring "Match substrings anywhere in word" *(correspondant à une sous-chaîne n'importe où dans le mot)*
S: regex "Match using regular expressions" *(correspondance en utilisant des expressions régulières)*
S: reverse "Match words given definition keywords" *(mots correspondant selon les définitions de mot clé)*
S: .
S: 250 Command complete

C: MATCH foldoc regex "s.si" *(chercher correspondance dans foldoc avec la stratégie regex)*

S: 152 7 matches found: list follows *(7 correspondances trouvées : la liste suit)*
S: foldoc Fast SCSI
S: foldoc SCSI
S: foldoc SCSI-1
S: foldoc SCSI-2
S: foldoc SCSI-3
S: foldoc Ultra-SCSI
S: foldoc Wide SCSI
S: .
S: 250 Command complete

C: MATCH wn substring "abcdefgh" *(chercher correspondance dans wn pour la sous-chaîne "abcdefgh")*
S: 552 No match *(pas de correspondance)*

8.3. Exemple 3 - Serveur en arrêt

C: [le client initialise la connexion]
S: 420 Server temporarily unavailable *(serveur temporairement indisponible)*

C: [le client initialise la connexion]
S: 421 Server shutting down at operator request *(serveur fermant à la demande de l'opérateur)*

8.4 Exemple 4 - Authentification

C: [le client initialise la connexion]
S: 220 dict.org dictd (version 0.9) <27831.860032493@dict.org>

C: SHOW DB *(montrer les bases de données)*

S: 110 1 database present: list follows *(une base de données présente : la liste suit)*
S: free "Free database" *(base de données libre)*
S: .
S: 250 Command complete

C: AUTH joesmith authentication-string

S: 230 Authentication successful *(authentification réussie)*

C: SHOW DB

S: 110 2 databases present: list follows
S: free "Free database"
S: licensed "Local licensed database"
S: .
S: 250 Command complete

9. Considérations pour la sécurité

La présente RFC ne soulève aucun problème de sécurité.

10. Références

[ASCII] "US-ASCII. Coded Character Set - 7-Bit American Standard Code for Information Interchange". Norme ANSI X3.4-1986, ANSI, 1986.

[FOLDOC] Howe, Denis, éd. "The Free On-Line Dictionary of Computing", <URL : http://wombat.doc.ic.ac.uk />

[ISO10646] Norme internationale ISO/CEI 10646-1:1993. "Technologies de l'information – Jeu de caractères universel codé sur plusieurs octets (UCS) -- Partie 1 : Architecture et plan de base multilingue". UTF-8 est décrit dans l'annexe R. UTF-16 est décrit à l'Annexe Q.

[JARGON] Le fichier en ligne du jargon du hacker, version 4.0.0, 25 JUL 1996, <URL : http://www.ccil.org/jargon/>

[KNUTH73] Knuth, Donald E. "The Art of Computer Programming", Tome 3 : Tri et recherche (Addison-Wesley éditeurs, 1973, pages 391 et 392). Knuth note que la méthode soundex a été décrite à l'origine par Margaret K. Odell et Robert C. Russell [Brevets US 1261167 (1918) et 1435663 (1922)].

[PZ85] Pollock, Joseph J. and Zamora, Antonio, "Automatic spelling correction in scientific and scholarly text," CACM, 27(4), avril 1985, 358-368.

[RFC0640] J. Postel, "Révision des codes de réponse FTP", juin 1974.

[RFC0821] J. Postel, "Protocole simple de transfert de messagerie", STD 10, août 1982.

[RFC0822] D. Crocker, "Norme pour le format des messages de texte de l'ARPA-Internet", STD 11, août 1982. (Obsolète, remplacée par la RFC5322)

[RFC0977] B. Kantor et P. Lapsley, ″Protocole de transfert des nouvelles du réseau", février 1986. (Obsolète, voir RFC3977)

[RFC2045] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 1 : Format des corps de message Internet", novembre 1996. *(D. S., MàJ par 2184, 2231, 5335.)*

[RFC1738] T. Berners-Lee et autres, "Localisateurs uniformes de ressource (URL)", décembre 1994. (P.S., Obsolète, voir les RFC4248 et 4266)

[RFC1760] N. Haller, "Système S/KEY de mot de passe à utilisation unique", février 1995. (Information)

[RFC1854] N. Freed, "Extension de service SMTP pour traitement en parallèle de commande", octobre 1995. (*Obsolète, voir* RFC2920, STD 60)

[RFC1939] J. Myers, M. Rose, "Protocole Post Office - version 3", mai 1996 . (*MàJ par* RFC1957, RFC2449) (STD0053)

[RFC2044] F. Yergeau, "UTF-8, un format de transformation d'Unicode et d'ISO 10646", octobre 1996. (*Obsolète, voir* RFC2279) *(Information)*

[RFC2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Protocole de transfert Hypertext -- HTTP/1.1", janvier 1997. (*Obsolète, voir* RFC2616) *(P.S.)*

[RFC2078] J. Linn, "Interface générique de programme d'application de service de sécurité, version 2", janvier 1997. *(Rendue obsolète par la RFC 2743.)*

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

[WEB1913] Webster's Revised Unabridged Dictionary (G & C. Merriam Co., 1913, édité par Noah Porter). Version en ligne préparée par MICRA, Inc., Plainfield, N.J. et éditée par Patrick Cassidy <cassidy@micra.com>. Pour plus d'informations, voir <URL:ftp://uiarchive.cso.uiuc.edu/pub/etext/gutenberg/etext96/pgw*>, et <URL:http://humanities.uchicago.edu/forms_unrest/webster.form.html>

[WORDNET] Miller, G.A. (1990), éd. "WordNet: An On-Line Lexical Database". Journal international de lexicographie. Volume 3, numéro 4. <URL:http://www.cogsci.princeton.edu/~wn/>

11. Remerciements

Merci à Arnt Gulbrandsen et à Nicolai Langfeldt pour de nombreuses discussions utiles. Merci à Bennet Yee, Doug

Hoffman, Kevin Martin, et Jay Kominek pour les essais extensifs et leurs retours sur les mises en œuvre initiales du serveur DICT. Merci à Zhong Shao pour ses conseils et son soutien.

Merci à Brian Kanto, Phil Lapsley et Jon Postel pour leur rédaction des RFC exemplaires qui ont été consultées durant la préparation de ce document.

Merci à Harald T. Alvestrand, dont les commentaires ont aidé à améliorer ce document.

12. Adresse des auteurs

Rickard E. Faith
mél : faith@cs.unc.edu (or faith@acm.org)

Bret Martin
mél : bamartin@miranda.org

La plus grande partie de ce travail a été réalisée alors que Bret Martin était étudiant à la Yale University.

13. Déclaration complète de droits de reproduction