Groupe de travail Réseau
Request for Comments : 2812
RFC mise à jour : 1459
Catégorie : Information

C. Kalt
avril 2000

Traduction Claude Brière de L'Isle

Relais pour la causette Internet : protocole client

Statut de ce mémoire
Le présent mémoire fournit des informations pour la communauté de l'Internet. Il ne spécifie aucune norme de l'Internet. La distribution du présent mémoire n'est soumise à aucune restriction.
(La présente traduction incorpore les errata 384, 385, 386, 991, 2306, 2821, 2822, 2991, 3001, 3246, et 3255)

Note de l'IESG :
Le protocole IRC lui-même permet plusieurs possibilités de transférer des données entre clients, et tout comme les autres mécanismes de transfert du style de la messagerie électronique, le receveur des données doit faire attention à la façon dont les données sont traitées. Pour plus d'informations sur les problèmes de sécurité posés par le protocole IRC, voir, par exemple http://www.irchelp.org/irchelp/security/ .

Résumé
Le protocole de relais de causette de l'Internet (IRC, *Internet Relay Chat*) est à utiliser avec la conférence fondée sur le texte, le plus simple client étant toute prise programmable capable de se connecter au serveur.

Le présent document définit le protocole client, et suppose que le lecteur est familiarisé avec l'architecture IRC [RFC2810].

Dans le présent document, les mots clés "DOIT", "NE DOIT PAS", "EXIGÉ", "DEVRA" NE DEVRA PAS", "DEVRAIT" NE DEVRAIT PAS", "RECOMMANDÉ", "PEUT", et "FACULTATIF" sont à interpréter comme décrit dans la [RFC2119].

Table des matières

1. Étiquettes

Cette section définit les identifiants utilisés pour les divers composants du protocole IRC.

1.1 Serveurs

Les serveurs sont identifiés de façon univoque par leur nom, qui a une longueur maximum de soixante trois (63) caractères. Voir les règles de grammaire du protocole (paragraphe 2.3.1) pour savoir ce qui peut être ou non utilisé dans un nom de serveur.

1.2 Clients

Pour chaque client, tous les serveurs DOIVENT avoir les informations suivantes : un identifiant univoque à l'échelle du réseau (dont le format dépend du type de client) et le serveur qui a introduit le client.

1.2.1 Utilisateurs

Chaque utilisateur se distingue des autres utilisateurs par un pseudonyme univoque qui a une longueur maximum de neuf (9) caractères. Voir dans les règles de grammaire du protocole (paragraphe 2.3.1) ce qui peut ou non être utilisé dans un pseudonyme.

Bien que la longueur maximum soit limitée à neuf caractères, les clients DEVRAIENT accepter de plus longues chaînes car elles pourraient être utilisées dans de futures évolutions du protocole.

1.2.1.1 Opérateurs

Pour permettre de conserver un minimum d'ordre raisonnable au sein du réseau IRC, une classe spéciale d'utilisateurs (les opérateurs) est autorisée à effectuer les fonctions générales de maintenance sur le réseau. Bien que les pouvoirs accordés à un opérateur puissent être considérés comme 'dangereux', ils sont néanmoins souvent nécessaires. Les opérateurs DEVRAIENT être capables d'effectuer des tâches de base sur le réseau telles que de déconnecter et reconnecter les serveurs en tant que de besoin. En reconnaissance de ce besoin, le protocole qu'on expose ici ne s'occupe des opérateurs qu'à l'égard de leur capacité à effectuer de telles fonctions. Voir les paragraphes 3.1.8 (SQUIT) et 3.4.7 (CONNECT).

Un pouvoir plus controversé des opérateurs est la capacité de retirer "de force" un utilisateur du réseau connecté, c'est-à-dire que les opérateurs sont capables de fermer la connexion entre tout client et serveur. La justification de cela est très délicate car son abus est à la fois destructeur et fâcheux, et son bénéfice proche d'inexistant. Pour les détails sur ce type d'action, voir au paragraphe 3.7.1 (KILL).

1.2.2 Services

Chaque service se distingue des autres services par un nom de service composé d'un pseudonyme et d'un nom de serveur. Comme pour les utilisateurs, le pseudonyme a une longueur maximum de neuf (9) caractères. Voir dans les règles de grammaire du protocole (paragraphe2.3.1) ce qui peut ou non être utilisé dans un pseudonyme.

1.3 Canaux

Les noms de canal sont des chaînes (commençant par un caractère '&', '#', '+' ou '!') d'une longueur allant jusqu'à cinquante (50) caractères. À part l'exigence que le premier caractère soit '&', '#', '+' ou '!', la seule restriction à un nom de canal est qu'il NE DEVRA PAS contenir d'espace (' '), de contrôle G (^G ou ASCII 7) ni de virgule (','). L'espace est utilisée comme séparateur de paramètre et la virgule est utilisée comme séparateur d'élément de liste par le protocole. Un caractère deux-points (':') peut aussi être utilisé comme délimiteur pour le gabarit de canal. Les noms de canal sont insensibles à la casse.

Voir dans les règles de grammaire du protocole (paragraphe 2.3.1) la syntaxe exacte d'un nom de canal.

Chaque préfixe caractérise un type différent de canal. La définition du type de canal n'est pas pertinente pour le protocole client-serveur et sort donc du domaine d'application du présent document. Les précisions figurent dans la [RFC2811] "Relais pour la causette Internet : gestion de canal".

2. Spécification du client IRC

2.1 Généralités

Le protocole décrit ici est à utiliser pour les seules connexions de client à serveur lorsque le client s'enregistre comme utilisateur.

2.2 Codes de caractères

Aucun jeu de caractères spécifique n'est spécifié. Le protocole se fonde sur un jeu de codes qui sont composés de huit (8) bits, constituant un octet. Chaque message peut être composé d'un nombre quelconque de ces octets ; cependant, certaines valeurs d'octet sont utilisées pour des codes de contrôle, qui agissent comme des délimiteurs de message.

Indépendamment d'être un protocole à 8 bits, les délimiteurs et les mots-clés sont tels que le protocole est principalement utilisable à partir de terminaux US-ASCII et d'une connexion telnet.

À cause de l'origine scandinave de l'IRC, les caractères {}|^ sont considérés respectivement comme l'équivalent en minuscules des caractères []\~. Ceci est d'une importance critique pour la détermination de l'équivalence de deux pseudonymes ou de deux noms de canal.

2.3 Messages

Serveurs et clients s'envoient chacun des messages, qui peuvent ou non générer une réponse. Si le message contient une commande valide, comme décrit dans les paragraphes suivants, le client devrait s'attendre à la réponse spécifiée mais il ne lui est pas conseillé d'attendre indéfiniment cette réponse ; la communication de client à serveur et de serveur à serveur est essentiellement asynchrone par nature.

Chaque message IRC peut consister en jusqu'à trois parties principales : le préfixe (FACULTATIF), la commande, et les paramètres de commande (maximum de quinze (15)). Le préfixe, la commande, et tous les paramètres sont chacun séparés par un caractère ASCII espace (0x20).

La présence d'un préfixe est indiquée par un seul caractère ASCII deux-points (':', 0x3a) en tête, qui DOIT être le premier caractère du message lui-même. Il DOIT n'y avoir AUCUN trou (espace) entre le deux-points et le préfixe. Le préfixe est utilisé par les serveurs pour indiquer la vraie origine du message. Si le préfixe manque dans le message, il est supposé avoir été généré sur la connexion d'où il a été reçu. Les clients NE DEVRAIENT PAS utiliser un préfixe lors de l'envoi d'un message ; si ils en utilisent un, le seul préfixe valide est le pseudonyme enregistré associé au client.

La commande DOIT être une commande IRC valide ou un nombre de trois (3) chiffres représentés en texte ASCII.

Les messages IRC sont toujours des lignes de caractères terminées par une paire retour chariot-saut à la ligne (CR-LF, *Carriage Return - Line Feed)* et ces messages NE DEVRONT PAS excéder 512 caractères, en comptant tous les caractères y compris le CR-LF de queue. Donc, 510 caractères maximum sont alloués pour la commande et ses paramètres. Il n'y a aucune disposition pour la continuation des lignes du message. Voir à la section 6 des précisions sur les mises en œuvre actuelles.

2.3.1 Format de message en BNF augmenté

Les messages du protocole doivent être extraits du flux d'octets contigu. La solution actuelle est de désigner deux

caractères, CR et LF, comme séparateurs de message. Les messages vides sont ignorés en silence, ce qui permet d'utiliser la séquence CR-LF entre les messages sans problème supplémentaire.

Le message extrait est analysé dans ses composants <préfixe>, <commande> et liste de paramètres (<params>).

La représentation en BNF augmenté en est :

```
message     =  [ ":" préfixe ESPACE ] commande [ params ] crlf
préfixe     =  nom_de_serveur / ( pseudonyme [ [ "!" utilisateur ] "@" hôte ] )
commande    =  1*lettre / 3chiffre
params      =  *14( ESPACE milieu ) [ ESPACE ":" en_queue ] =/ 14( ESPACE milieu ) [ ESPACE [ ":" ] en_queue ]
nospcrlfcl  =  %x01-09 / %x0B-0C / %x0E-1F / %x21-39 / %x3B-FF          ; tout octet sauf NUL, CR, LF, " " et ":"
milieu      =  nospcrlfcl *( ":" / nospcrlfcl )
en_queue    =  *( ":" / " " / nospcrlfcl )
ESPACE      =  %x20                                 ; caractère espace
crlf        =  %x0D %x0A                            ; "retour chariot" "saut à la ligne"
```

Notes :

1) Après l'extraction de la liste des paramètres, tous les paramètres sont égaux qu'ils correspondent à <milieu> ou à <en_queue>. <en_queue> est juste un tour syntaxique pour permettre une ESPACE au sein du paramètre.

2) Le caractère NUL (%x00) n'est pas un caractère spécial dams la trame de message, et pourrait bien se trouver à l'intérieur d'un paramètre, mais cela causerait une complexité supplémentaire dans le traitement normal de chaîne C. Donc, NUL n'est pas admis au sein des messages.

La plupart des messages du protocole spécifient de la sémantique et de la syntaxe supplémentaire pour les chaînes de paramètre extraites en fonction de leur position dans la liste. Par exemple, de nombreuses commandes de serveur vont supposer que le premier paramètre après la commande est la liste des cibles, ce qui peut être décrit par :

```
cible       =  pseudonyme / serveur
msgcible    =  msgto *( "," msgto )
msgto       =  canal / ( utilisateur [ "%" hôte ] "@" nom_serveur )
msgto       =/ ( utilisateur "%" hôte ) / chiffre
msgto       =/ pseudonyme / ( pseudonyme "!" utilisateur "@" hôte )
canal       =  ( "#" / "+" / ( "!" canalid ) / "&" ) canalchaîne [ ":" canalchaîne ]
nom_serveur =  nom_hôte
hôte        =  nom_hôte / adrhôte
nom_hôte    =  petitnom *( "." petitnom )
petitnom    =  ( lettre / chiffre ) [ *( lettre / chiffre / "-" ) *( lettre / chiffre ) ]     ; comme spécifié dans la [RFC1123]
adrhôte     =  ip4adr / ip6adr
ip4adr      =  1*3chiffre "." 1*3chiffre "." 1*3chiffre "." 1*3chiffre
ip6adr      =  1*chiffrehex 7( ":" 1*chiffrehex )
ip6adr      =/ "0:0:0:0:0:" ( "0" / "FFFF" ) ":" ip4adr
pseudonyme  =  ( lettre / special ) *8( lettre / chiffre / special / "-" )
gabaritcible =  ( "$" / "#" ) gabarit                ; voir des précisions sur les gabarits admis au paragraphe 3.3.1
canalchaîne =  %x01-06 / %x08-09 / %x0B-0C / %x0E-1F / %x21-2B
canalchaîne =/ %x2D-39 / %x3B-FF                     ; tout octet sauf NUL, BELL, CR, LF, " ", "," et ":"
canalid     =  5( %x41-5A / chiffre )                ; 5( A-Z / 0-9 )
```

Les autres règles de syntaxe de paramètres sont :

```
utilisateur =  1*( %x01-09 / %x0B-0C / %x0E-1F / %x21-3F / %x41-FF )
                                                     ; tout octet sauf NUL, CR, LF, " " et "@"
clé         =  1*23( %x01-05 / %x07-08 / %x0C / %x0E-1F / %x21-7F )
                                  ; tout caractère US_ASCII de 7 bits, sauf NUL, CR, LF, FF, h/v TAB, et " "
lettre      =  %x41-5A / %x61-7A                     ; A-Z / a-z
chiffre     =  %x30-39                               ; 0 à 9
chiffrehex  =  chiffre / "A" / "B" / "C" / "D" / "E" / "F"
special     =  %x5B-60 / %x7B-7D ; "[", "]", "\", "`", "_", "^", "{", "|", "}"
```

Notes :

1) La syntaxe de <adrhôte> n'est donnée ici que dans le but d'indiquer le format à suivre pour les adresses IP. Cela reflète le fait que les seules mises en œuvre disponibles de ce protocole utilisent TCP/IP comme protocole réseau sous-jacent mais cela n'est pas destiné à empêcher l'utilisation d'autres protocoles.
2) <nom_hôte> a une longueur maximum de 63 caractères. C'est une limitation du protocole car les noms d'hôte Internet

(en particulier) peuvent être plus longs. Une telle restriction est nécessaire parce que les messages IRC sont limités à 512 caractères. Les clients qui se connectent à partir d'un hôte dont le nom fait plus de 63 caractères sont enregistrés en utilisant l'adresse (numérique) de l'hôte au lieu du nom de l'hôte.

3) Certains paramètres utilisés dans les paragraphes suivants de ce document ne sont pas définis ici car il n'y a rien de spécifique à leur sujet en dehors du nom qui est utilisé par convention. Ces paramètres suivent la syntaxe générale définie pour <params>.

2.4 Réponses numériques

La plupart des messages envoyés au serveur génèrent une réponse d'une certaine forme. La réponse la plus courante est la réponse numérique, utilisée à la fois pour les erreurs et pour les réponses normales. La réponse numérique DOIT être envoyée comme un message consistant en le préfixe de l'envoyeur, le numéro à trois chiffres, et la cible de la réponse. Une réponse numérique n'est pas admise si elle a un client pour origine. Sous tous les autres points de vue, une réponse numérique est tout comme un message normal, sauf que le mot clé est constitué de numéros de trois chiffres plutôt que d'une chaîne de lettres. Une liste des différentes réponses est fournie à la Section 5 (Réponses).

2.5 Expressions de caractères génériques

Lorsque les caractères génériques sont autorisés dans une chaîne, on les appelle un "gabarit".

Pour les besoins de la correspondance de chaînes, le protocole permet l'utilisation de caractères spéciaux : '?' (%x3F) pour correspondre à un seul et unique caractère, et '*' (%x2A) pour correspondre à un nombre quelconque de caractères. Ces deux caractères peuvent être esquivés en utilisant le caractère '\' (%x5C).

La syntaxe ABNF pour cela est :

```
gabarit     =  *( nowild / noesc wildone / noesc wildmany )
wildone     =  %x3F
wildmany    =  %x2A
nowild      =  %x01-29 / %x2B-3E / %x40-FF      ; tout octet sauf NUL, "*", "?"
noesc       =  %x01-5B / %x5D-FF                ; tout octet sauf NUL et "\"
matchone    =  %x01-FF                          ; correspond à wildone
matchmany   =  *matchone                        ; correspond à wildmany
```

Exemples :

```
a?c         ; correspond à toute chaîne de 3 caractères commençant par "a" et se terminant par "c"
a*c         ; correspond à toute chaîne d'au moins 2 caractères commençant par "a" et se terminant par "c"
```

3. Détails du message

Les pages qui suivent contiennent la description de chacun des message reconnus par le serveur et client IRC. Toutes les commandes décrites dans cette section DOIVENT être mises en œuvre par tout serveur conforme au présent protocole.

Lorsque la réponse ERR_NOSUCHSERVER est retournée, cela signifie que la cible du message n'a pas pu être trouvée. Le serveur NE DOIT PAS envoyer d'autres réponses après cette erreur pour cette commande.

Le serveur auquel un client est connecté est obligé d'analyser le message complet, et de retourner toute erreur appropriée.

Si plusieurs paramètres sont présentés, la validité de chacun d'eux DOIT alors être vérifiée et les réponses appropriées DOIVENT être renvoyées au client. Dans le cas de messages incorrects qui utilisent des listes de paramètres avec des virgules comme séparateur d'élément, une réponse DOIT être envoyée pour chaque élément.

3.1 Enregistrement de connexion

Les commandes décrites ici sont utilisées pour enregistrer comme utilisateur une connexion avec un serveur IRC ainsi que pour se déconnecter correctement.

Une commande "PASS" n'est pas exigée pour qu'une connexion client soit enregistrée, mais elle DOIT précéder la dernière de la combinaison de NICK/USER (pour une connexion d'utilisateur) ou de la commande SERVICE (pour une connexion de service). L'ordre RECOMMANDÉ pour l'enregistrement d'un client est le suivant :

1. Message Pass
2. Message Nick 2. Message Service
3. Message User

En cas de réussite, le client va recevoir un message RPL_WELCOME (pour un utilisateurs) ou RPL_YOURESERVICE (pour un service) qui indique que la connexion est maintenant enregistrée et connue de tout le réseau IRC. Le message de réponse DOIT contenir l'identifiant complet du client avec lequel il s'est enregistré.

3.1.1 Message de mot de passe

Commande : PASS
Paramètres : <mot de passe>

La commande PASS est utilisée pour établir un 'mot de passe de connexion'. Le mot de passe est facultatif et DOIT être réglé avant toute tentative d'enregistrer la connexion. Actuellement, cela exige que l'utilisateur envoie une commande PASS avant d'envoyer la combinaison NICK/USER.

Réponses numériques :
ERR_NEEDMOREPARAMS ERR_ALREADYREGISTRED

Exemple :
PASS secretpasswordhere

3.1.2 Message Nick

Commande : NICK
Paramètres : <pseudonyme>

La commande NICK est utilisée pour donner à l'utilisateur un pseudonyme ou changer celui qui existe.

Réponses numériques :
ERR_NONICKNAMEGIVEN ERR_ERRONEUSNICKNAME
ERR_NICKNAMEINUSE ERR_NICKCOLLISION
ERR_UNAVAILRESOURCE ERR_RESTRICTED

Exemples :
NICK Wiz ; Introduit un nouveau pseudonyme "Wiz" si la session n'est pas encore enregistrée, ou l'utilisateur change son pseudonyme en "Wiz"

:WiZ!jto@tolsun.oulu.fi NICK Kilroy ; Le serveur dit que WiZ a changé son pseudonyme en Kilroy.

3.1.3 Message User

Commande : USER
Paramètres : <utilisateur> <mode> <non utilisé> <nom réel>

La commande USER est utilisée au commencement de la connexion pour spécifier le nom d'utilisateur, le nom d'hôte et le nom réel d'un nouvel utilisateur.

Le paramètre <mode> devrait être un numéro, et il peut être utilisé pour régler automatiquement les modes d'utilisateur lors de l'enregistrement auprès du serveur. Ce paramètre est un gabarit binaire, avec seulement 2 bits qui ont une signification : si le bit 2 est établi *(à 1)*, le mode d'utilisateur 'w' sera établi et si le bit 3 est établi, le mode d'utilisateur 'i' sera établi. (Voir au paragraphe 3.1.5 "Message Mode d'utilisateur").

Le <nom réel> peut contenir des caractères espace.

Réponses numériques :
ERR_NEEDMOREPARAMS ERR_ALREADYREGISTRED

Exemple :
USER guest 0 * :Ronnie Reagan ; L'usager s'enregistre lui-même avec un nom d'utilisateur de "guest" et un nom réel de "Ronnie Reagan".

USER guest 8 * :Ronnie Reagan ; L'usager s'enregistre avec un nom d'utilisateur de "guest" et le nom réel "Ronnie Reagan", et demande que ce soit invisible.

3.1.4 Message Oper

Commande : OPER
Paramètres : <nom> <mot de passe>

Un utilisateur normal utilise la commande OPER pour obtenir les privilèges d'opérateur. La combinaison de <nom> et de <mot de passe> est EXIGÉE pour obtenir les privilèges d'opérateur. En cas de succès, l'utilisateur recevra un message MODE (voir au paragraphe 3.1.5) indiquant les nouveaux modes d'utilisateur.

Réponses numériques :
ERR_NEEDMOREPARAMS RPL_YOUREOPER
ERR_NOOPERHOST ERR_PASSWDMISMATCH

Exemple :
OPER foo bar ; tentative de s'enregistrer comme opérateur en utilisant un nom d'utilisateur de "foo" et "bar" comme mot de passe.

3.1.5 Message Mode d'utilisateur

Commande : MODE
Paramètres : <pseudonyme> *(("+" / "-") *("i" / "w" / "o" / "O" / "r"))

Les modes d'utilisateur sont normalement des changements qui affectent la façon dont le client est vu par les autres ou quels messages supplémentaires sont envoyés au client.

Une commande MODE d'utilisateur DOIT n'être acceptée que si l'envoyeur du message et le pseudonyme donné comme paramètre sont le même. Si aucun autre paramètre n'est donné, le serveur va alors retourner les réglages actuels pour le pseudonyme.

Les modes disponibles sont les suivants :
a – l'utilisateur est marqué comme absent ;
i – l'utilisateur est marqué comme invisible ;
w – l'utilisateur reçoit un avertissement ;
r – apporte des restrictions à la connexion de l'utilisateur ;
o – fanion d'opérateur ;
O – fanion d'opérateur local ;
s – marque un utilisateur pour la réception de notices du serveur.

Des modes supplémentaires pourraient devenir disponibles ultérieurement.

Le fanion 'a' NE DEVRA PAS être modifié par l'utilisateur en utilisant la commande MODE. L'utilisation de la commande AWAY est EXIGÉE.

Si un utilisateur tente de se changer lui-même en opérateur en utilisant le fanion "+o" ou "+O", la tentative DEVRAIT être ignorée car les utilisateurs pourraient outrepasser les mécanismes d'authentification de la commande OPER. Il n'y a pas cependant de restriction sur le fait que quiconque se défasse lui-même du statut d'opérateur (en utilisant "-o" ou "-O").

D'un autre côté, si un utilisateur tente de supprimer lui-même toutes les restrictions qui l'affectent en utilisant le fanion "-r", la tentative DEVRAIT être ignorée. Il n'y a cependant pas de restriction à ce que quiconque se défasse de ses privilèges (en utilisant "+r"). Ce fanion est normalement établi par le serveur au moment de la connexion pour des raisons administratives. Bien que les restrictions imposées soient au choix de la mise en œuvre, un utilisateur restreint ne sera normalement pas autorisé à changer de pseudonyme, ni à faire usage du statut d'opérateur du canal sur les canaux.

Le fanion 's' est obsolète mais PEUT encore être utilisé.

Réponses numériques :
ERR_NEEDMOREPARAMS ERR_USERSDONTMATCH
ERR_UMODEUNKNOWNFLAG RPL_UMODEIS

Exemples :
MODE WiZ –w ; Commande de WiZ pour arrêter la réception des messages WALLOPS.
MODE Angel +i ; Commande de Angel pour se rendre invisible.
MODE WiZ –o ; WiZ se “déoppe“ (se retire le statut d’opérateur).

3.1.6 Message Service

Commande : SERVICE
Paramètres : <pseudonyme> <réservé> <distribution> <type> <réservé> <info>

La commande SERVICE est utilisée pour enregistrer un nouveau service. Les paramètres de la commande spécifient pour un nouveau service le pseudonyme du service, sa distribution, son type et ses informations.

Le paramètre <distribution> est utilisé pour spécifier la visibilité d’un service. Le service peut n’être connu que des serveurs qui ont un nom qui correspond à la distribution. Pour qu’un serveur correspondant ait connaissance du service, le chemin réseau entre ce serveur et le serveur sur lequel le service est connecté DOIT être composé de serveurs dont les noms correspondent tous au gabarit.

Le paramètre <type> est actuellement réservé pour une utilisation future.

Réponses numériques :
ERR_ALREADYREGISTRED ERR_NEEDMOREPARAMS
ERR_ERRONEUSNICKNAME
RPL_YOURESERVICE RPL_YOURHOST
RPL_MYINFO

Exemple :
SERVICE dict * *.fr 0 0 :French Dictionary ; Service s’enregistrant sous le nom de "dict". Ce service ne sera disponible que sur les serveurs dont le nom correspond à "*.fr".

3.1.7 Message Quit

Commande : QUIT
Paramètres : [<Quit Message>]

Une session client se termine par un message quit. Le serveur en accuse réception par l’envoi au client d’un message ERROR.

Réponse numérique : Aucune

Exemples :
QUIT :Parti déjeuner ; format de message préféré.
:syrk!kalt@millennium.stealth.net QUIT :Parti déjeuner ; l’usager syrk a quitté IRC pour aller déjeuner.

3.1.8 Message Squit

Commande : SQUIT
Paramètres : <serveur> <commentaire>

La commande SQUIT n’est disponible qu’aux opérateurs. Elle est utilisée pour déconnecter les liaisons de serveur. Les serveurs peuvent aussi générer des messages SQUIT dans des conditions d’erreur. Un message SQUIT peut aussi cibler une connexion de serveur distant. Dans ce cas, le message SQUIT va simplement être envoyé au serveur distant sans affecter les serveurs situés entre l’opérateur et le serveur distant.

Le <commentaire> DEVRAIT être fourni par tous les opérateurs qui exécutent une commande SQUIT pour un serveur distant. Le serveur auquel il est ordonné de déconnecter son homologue génère un message WALLOPS avec un <commentaire> inclus, de sorte que les autres utilisateurs puissent être informés de la raison de cette action.

Réponses numériques :
ERR_NOPRIVILEGES ERR_NOSUCHSERVER ERR_NEEDMOREPARAMS

Exemples :
SQUIT tolsun.oulu.fi :mauvaise liaison ? ; Commande au serveur tolson.oulu.fi de terminer sa connexion, avec le commentaire "mauvaise liaison".
:Trillian SQUIT cm22.eng.umd.edu :Serveur hors de contrôle ; Commande de Trillian de déconnecter du réseau "cm22.eng.umd.edu" avec le commentaire "Serveur hors de contrôle".

3.2 Opérations du canal

Ce groupe de messages concerne la manipulation des canaux, de leurs propriétés (les modes de canal) et leur contenu (en fait les utilisateurs). Pour cette raison, ces messages NE DEVRONT PAS être disponibles aux services.

Tous ces messages sont des demandes qui seront accordées ou non par le serveur. Le serveur DOIT envoyer une réponse qui informe l'utilisateur de la satisfaction de la demande, de son refus, ou génère une erreur. Lorsque le serveur satisfait la demande, le message est normalement renvoyé (éventuellement reformaté) à l'utilisateur avec le préfixe réglé à l'utilisateur lui-même.

Les règles qui gouvernent la façon dont sont gérés les canaux sont mises en application par les serveurs. Ces règles sortent du domaine d'application du présent document. Des précisions sont fournies dans la [RRC2811] "Relais de la causette Internet : gestion de canal".

3.2.1 Message Join

Commande : JOIN
Paramètres : (<canal> *("," <canal>) [<clé> *("," <clé>)]) / "0"

La commande JOIN est utilisée par un utilisateur pour demander à commencer d'écouter sur le canal spécifié. Les serveurs DOIVENT être capables d'analyser les arguments sous la forme d'une liste de cibles, mais NE DEVRAIENT PAS utiliser les listes lors de l'envoi de messages JOIN aux clients.

Une fois qu'un utilisateur s'est joint à un canal, il reçoit des informations sur toutes les commandes que reçoit son serveur et qui affectent le canal. Cela inclut les commandes JOIN, MODE, KICK, PART, QUIT et bien sûr PRIVMSG/NOTICE. Cela permet aux membres du canal d'être informés des autres membres du canal, ainsi que des modes du canal.

Si une commande JOIN réussit, l'utilisateur reçoit un message JOIN en confirmation et on lui envoie ensuite les sujets *(topic)* du canal (en utilisant RPL_TOPIC) et la liste des utilisateurs qui sont sur le canal (avec RPL_NAMREPLY) qui DOIT inclure l'utilisateur qui vient de s'y joindre.

Noter que ce message accepte un argument ("0") particulier, qui est une demande spéciale de quitter tous les canaux dont l'utilisateur est actuellement membre. Le serveur va traiter ce message comme si l'utilisateur avait envoyé une commande PART (voir au paragraphe 3.2.2) pour chaque canal dont il est membre.

Réponses numériques :
ERR_NEEDMOREPARAMS ERR_BANNEDFROMCHAN
ERR_INVITEONLYCHAN ERR_BADCHANNELKEY
ERR_CHANNELISFULL ERR_BADCHANMASK
ERR_NOSUCHCHANNEL ERR_TOOMANYCHANNELS
ERR_TOOMANYTARGETS ERR_UNAVAILRESOURCE
RPL_TOPIC

Exemples :
JOIN #foobar ; Commande pour se joindre au canal #foobar.
JOIN &foo fubar ; Commande pour se joindre au canal &foo en utilisant la clé "fubar".
JOIN #foo,&bar fubar ; Commande pour se joindre au canal #foo avec la clé "fubar" et &bar sans clé.
JOIN #foo,#bar fubar,foobar ; Commande pour se joindre au canal #foo avec la clé "fubar", et au canal #bar avec la clé "foobar".
JOIN #foo,#bar ; Commande pour se joindre aux canaux #foo et #bar.
JOIN 0 ; Quitter tous les canaux actuellement rejoins.
:WiZ!jto@tolsun.oulu.fi JOIN #Twilight_zone ; message JOIN de WiZ sur le canal #Twilight_zone

3.2.2 Message Part

Commande : PART
Paramètres : <canal> *("," <canal>) [<Message Part>]

La commande PART cause l'envoi par l'utilisateur du message à retirer de la liste des membres actifs pour tous les canaux figurant sur la liste de la chaîne de paramètres. Si un "Message Part" est donné, il sera envoyé à la place du message par défaut, le pseudonyme. Cette demande est toujours accordée par le serveur.

Les serveurs DOIVENT être capables d'analyser les arguments sous la forme d'une liste de cibles, mais ils NE DEVRAIENT PAS utiliser les listes lors de l'envoi de messages PART aux clients.

Réponses numériques :
ERR_NEEDMOREPARAMS ERR_NOSUCHCHANNEL
ERR_NOTONCHANNEL

Exemples :
PART #twilight_zone ; Commande de quitter le canal "#twilight_zone"
PART #oz-ops,&group5 ; Commande de quitter les deux canaux "&group5" et "#oz-ops".
:WiZ!jto@tolsun.oulu.fi PART #playzone :I lost ; L'usager WiZ quitte le canal "#playzone" avec le message "I lost".

3.2.3 Message Mode de canal

Commande : MODE
Paramètres : <canal> *(("-" / "+") *<modes> *<modeparams>)

La commande MODE est fournie de façon à ce que les utilisateurs puissent interroger et changer les caractéristiques d'un canal. Pour des précisions sur les modes disponibles et leur utilisation, voir la [RFC2811] "Relais de causette Internet : gestion de canal". Noter qu'il y a une limite maximum de trois (3) changements par commande pour les modes qui prennent un paramètre.

Réponses numériques :
ERR_NEEDMOREPARAMS ERR_KEYSET
ERR_NOCHANMODES ERR_CHANOPRIVSNEEDED
ERR_USERNOTINCHANNEL ERR_UNKNOWNMODE
RPL_CHANNELMODEIS ERR_NOSUCHCHANNEL
RPL_BANLIST RPL_ENDOFBANLIST
RPL_EXCEPTLIST RPL_ENDOFEXCEPTLIST
RPL_INVITELIST RPL_ENDOFINVITELIST
RPL_UNIQOPIS

Les exemples qui suivent sont données pour aider à comprendre la syntaxe de la commande MODE, mais se réfèrent aux modes définis dans la [RFC2811] "Relais de causette Internet : gestion de canal".

Exemples :
MODE #Finnish +imI *!*@*.fi ; Commande pour rendre modéré le canal #Finnish et 'seulement sur invitation' avec des utilisateurs ayant un nom d'hôte correspondant à *.fi invités automatiquement.
MODE #Finnish +o Kilroy ; Commande pour donner les privilèges 'chanop' à Kilroy sur le canal #Finnish.
MODE #Finnish +v Wiz ; Commande pour permettre à WiZ de parler sur #Finnish.
MODE #Fins –s ; Commande pour retirer le fanion 'secret' du canal #Fins.
MODE #42 +k oulu ; Commande pour régler la clé de canal à "oulu".
MODE #42 -k oulu ; Commande pour retirer la clé de canal "oulu" sur le canal "#42".
MODE #eu-opers +l 10 ; Commande pour régler à 10 la limite du nombre d'utilisateurs sur le canal "#eu-opers".
:WiZ!jto@tolsun.oulu.fi MODE #eu-opers –l ; L'utilisateur "WiZ" retire la limite du nombre d'utilisateurs sur le canal "#eu-opers".
MODE &oulu +b ; Commande pour faire la liste des gabarits d'interdiction réglés pour le canal "&oulu".
MODE &oulu +b *!*@* ; Commande pour empêcher tous les utilisateurs de se joindre à "&oulu".
MODE &oulu +b *!*@*.edu +e *!*@*.bu.edu ; Commande pour empêcher tout utilisateur avec un nom d'hôte correspondant à *.edu de se joindre, sauf si il correspond à *.bu.edu
MODE #bu +be *!*@*.edu *!*@*.bu.edu ; Commentaire pour empêcher tout utilisateur avec un nom d'hôte correspondant à *.edu de se joindre, sauf si il correspond à *.bu.edu
MODE #meditation e ; Commande pour faire la liste des gabarits d'exception réglés pour le canal "#meditation".
MODE #meditation I ; Commande pour faire la liste des gabarits d'invitations établis pour le canal "#meditation".

MODE !12345ircd O ; Commande pour demander qui est le créateur de canal pour "!12345ircd".

3.2.4 Message Topic

Commande : TOPIC
Paramètres : <canal> [<sujet>]
La commande TOPIC est utilisée pour changer/ouvrir le sujet d'un canal. Le sujet pour le canal <canal> est retourné si il n'y a pas de <sujet> donné. Si le paramètre <sujet> est présent, le sujet de ce canal sera changé, si cette action est permise à l'utilisateur qui la demande. Si le paramètre <sujet> est une chaîne vide, le sujet de ce canal sera supprimé.

Réponses numériques :
ERR_NEEDMOREPARAMS ERR_NOTONCHANNEL
RPL_NOTOPIC RPL_TOPIC
ERR_CHANOPRIVSNEEDED ERR_NOCHANMODES
ERR_NOSUCHCHANNEL

Exemples :
:WiZ!jto@tolsun.oulu.fi TOPIC #test :New topic ; L'utilisateur Wiz établit le sujet.
TOPIC #test :another topic ; Commande pour régler le sujet sur #test à "another topic".TOPIC
#test : ; Commande pour supprimer le sujet sur #test.
TOPIC #test ; Commande pour vérifier le sujet sur #test.

3.2.5 Message Names

Commande : NAMES
Paramètres : [<canal> *("," <canal>) [<cible>]]

En utilisant la commande NAMES, un utilisateur peut faire la liste de tous les pseudonymes qui lui sont visibles. Pour des précision sur ce qui est visible et ce qui ne l'est pas, voir la [RFC2811] "Relais de causette Internet : gestion de canal". Le paramètre <canal> spécifie sur quel ou quels canaux retourner des informations. Il n'y a pas de réponse d'erreur pour de mauvais noms de canal.

Si aucun paramètre <canal> n'est donné, il est retourné une liste de tous les canaux et de leurs occupants. À la fin de cette liste, une liste des utilisateurs qui sont visibles mais soit ne sont sur aucun canal, soit sont sur un canal qui n'est pas visible, sont énumérés comme étant sur le `canal' "*".

Si le paramètre <cible> est spécifié, la demande est transmise au serveur qui va générer la réponse.

Les caractères génériques *(wildcards)* sont autorisés dans le paramètre <cible> .

Réponses numériques :
ERR_TOOMANYMATCHES ERR_NOSUCHSERVER
RPL_NAMREPLY RPL_ENDOFNAMES

Exemples :
NAMES #twilight_zone,#42 ; Commande pour faire la liste des utilisateurs visibles sur #twilight_zone et #42
NAMES ; Commande pour faire la liste de tous les canaux et utilisateurs visibles.

3.2.6 Message List

Commande : LIST
Paramètres : [<canal> *("," <canal>) [<cible>]]

La commande list est utilisée pour faire la liste des canaux et de leur sujet. Si le paramètre <canal> est utilisé, seul l'état de ces canaux est affiché.

Si le paramètre <cible> est spécifié, la demande est transmise au serveur qui va générer la réponse.

Les caractères génériques sont admis dans le paramètre <cible>.

Réponses numériques :
ERR_TOOMANYMATCHES ERR_NOSUCHSERVER

RPL_LIST RPL_LISTEND

Exemples :
LIST ; Commande pour faire la liste de tous les canaux.
LIST #twilight_zone,#42 ; Commande pour faire la liste des canaux #twilight_zone et #42

3.2.7 Message Invite

Commande : INVITE
Paramètres : <pseudonyme> <canal>

La commande INVITE est utilisée pour inviter un utilisateur sur un canal. Le paramètre <pseudonyme> est le pseudonyme de la personne à inviter sur le canal cible <canal>. Il n'y a pas d'exigence que le canal sur lequel l'utilisateur cible est invité doive exister ou soit un canal valide. Cependant, si le canal existe, seuls les membres du canal sont autorisés à inviter d'autres utilisateurs. Lorsque le canal a le fanion Seulement_sur_invitation établi, seuls les opérateurs de canal peuvent produire une commande INVITE.

Seuls l'utilisateur invitant et l'utilisateur invité vont recevoir la notification de l'invitation. Les autres membres du canal ne sont pas notifiés. (C'est différent des changements de MODE, et est à l'occasion une source de trouble pour les utilisateurs.)

Réponses numériques :
ERR_NEEDMOREPARAMS ERR_NOSUCHNICK
ERR_NOTONCHANNEL ERR_USERONCHANNEL
ERR_CHANOPRIVSNEEDED
RPL_INVITING RPL_AWAY

Exemples :
:Angel!wings@irc.org INVITE Wiz #Dust ; Message à WiZ quand il a été invité par l'utilisateur Angel sur le canal #Dust
INVITE Wiz #Twilight_Zone ; Commande pour inviter WiZ à #Twilight_zone

3.2.8 Commande Kick

Commande : KICK
Paramètres : <canal> *("," <canal>) <utilisateur> *("," <utilisateur>) [<commentaire>]

La commande KICK peut être utilisée pour demander le retrait forcé d'un utilisateur d'un canal. Elle cause le départ de force de <utilisateur> de <canal>. Pour que le message soit syntaxiquement correct, il DOIT y avoir un paramètre canal et plusieurs paramètres utilisateur, ou autant de paramètres canal qu'il y a de paramètres utilisateur. Si un "commentaire" est donné, il sera envoyé à la place du message par défaut qui est le pseudonyme de l'utilisateur produisant le KICK.

Le serveur NE DOIT PAS envoyer de message KICK avec plusieurs canaux ou utilisateurs aux clients. Cela est nécessaire pour conserver la rétro compatibilité avec les vieux logiciels client.

Réponses numériques :
ERR_NEEDMOREPARAMS ERR_NOSUCHCHANNEL
ERR_BADCHANMASK ERR_CHANOPRIVSNEEDED
ERR_USERNOTINCHANNEL ERR_NOTONCHANNEL

Exemples :
KICK &Melbourne Matthew ; Commande pour sortie Matthew de &Melbourne
KICK #Finnish John :Speaking English ; Commande pour sortir John de #Finnish avec "Speaking English" comme cause (commentaire).
:WiZ!jto@tolsun.oulu.fi KICK #Finnish John ; Message KICK sur le canal #Finnish de la part de WiZ pour retirer John du canal

3.3 Envoi des messages

Le principal objet du protocole IRC est de fournir une base aux clients pour communiquer les uns avec les autres. PRIVMSG, NOTICE et SQUERY (décrits au paragraphe 3.5 sur les interrogations et commandes de service) sont les seuls messages disponibles qui effectuent réellement la livraison d'un message de texte d'un client à un autre - le reste le rend juste possible et essaye d'assurer que cela se produit de façon fiable et structurée.

3.3.1 Messages Private

Commande : PRIVMSG
Paramètres : <msgtarget> <texte à envoyer>

PRIVMSG est utilisé pour envoyer des messages privés entre les utilisateurs, ainsi que pour envoyer des messages aux canaux. <msgtarget> est normalement le pseudonyme du receveur du message, ou un nom de canal.

Le paramètre <msgtarget> peut aussi être un gabarit d'hôte (#<gabarit>) ou un gabarit de serveur ($<gabarit>). Dans les deux cas, le serveur va seulement envoyer le PRIVMSG à ceux dont le serveur ou hôte correspond au gabarit. Le gabarit DOIT contenir au moins 1 (un) "." et pas de caractère générique suivant le dernier ".". Cette exigence existe pour empêcher les gens d'envoyer des messages à "#*" ou "$*", qui seraient diffusés à tous les utilisateurs. Les caractères génériques sont les caractères '*' et '?'. Cette extension de la commande PRIVMSG n'est disponible qu'aux opérateurs.

Réponses numériques :
ERR_NORECIPIENT ERR_NOTEXTTOSEND
ERR_CANNOTSENDTOCHAN ERR_NOTOPLEVEL
ERR_WILDTOPLEVEL ERR_TOOMANYTARGETS
ERR_NOSUCHNICK
RPL_AWAY

Exemples :
:Angel!wings@irc.org PRIVMSG Wiz :As tu reçu ce message ? ; Message de Angel à Wiz.
PRIVMSG Angel :oui, je l'ai reçu ! ; Commande pour envoyer un message à Angel.
PRIVMSG jto@tolsun.oulu.fi :Hello ! ; Commande pour envoyer un message à un utilisateur sur le serveur tolsun.oulu.fi avec le nom d'utilisateur de "jto".
PRIVMSG kalt%millennium.stealth.net@irc.stealth.net :Es-tu un grenouille ? ; Message à un utilisateur sur le serveur irc.stealth.net avec le nom d'utilisateur de "kalt", et connecté à partir de l'hôte millennium.stealth.net.
PRIVMSG kalt%millennium.stealth.net :Aimes tu le fromage ? ; Message à un utilisateur sur le serveur local avec le nom d'utilisateur "kalt", et connecté de l'hôte millennium.stealth.net.
PRIVMSG Wiz!jto@tolsun.oulu.fi :Hello ! ; Message à l'utilisateur avec le pseudonyme Wiz qui est connecté à partir de l'hôte tolsun.oulu.fi et a le nom d'utilisateur "jto".
PRIVMSG $*.fi :Le serveur tolsun.oulu.fi redémarre. ; Message à tous ceux dont le nom correspond à *.fi sur un serveur.
PRIVMSG #*.edu :NSFNet subit des travaux, s'attendre à des interruptions ; Message à tous les utilisateurs qui viennent d'un hôte dont le nom correspond avec *.edu.

3.3.2 Notice

Commande : NOTICE
Paramètres : <msgtarget> <texte>

La commande NOTICE est utilisée de la même façon que PRIVMSG. La différence entre NOTICE et PRIVMSG est que les réponses automatiques NE DOIVENT PAS être envoyées en réponse à un message NOTICE. Cette règle s'applique aussi aux serveurs – ils NE DOIVENT PAS renvoyer de réponse d'erreur au client à réception d'une notice. L'objet de cette règle est d'éviter des boucles entre clients qui envoient automatiquement quelque chose en réponse à ce qu'ils reçoivent.

Cette commande est disponible aussi bien aux services qu'aux utilisateurs.

Elle est normalement utilisée par les services, et les automates (clients avec un AI ou autre programme interactif qui contrôle leurs actions).

Voir au paragraphe précédent PRIVMSG les détails sur les réponses et les exemples.

3.4 Commandes d'interrogations de serveur

Le groupe de commandes d'interrogation de serveur a été conçu pour retourner des informations sur tout serveur qui est connecté au réseau.

Dans ces interrogations, où un paramètre apparaît comme <cible>, les gabarits de caractères génériques sont normalement

valides. Pour chaque paramètre, cependant, seule une interrogation et ensemble de réponses est à générer. Dans la plupart des cas, si un pseudonyme est donné, il va signifier le serveur auquel l'utilisateur est connecté.

Ces messages n'ont normalement que peu de valeur pour les services, il est donc RECOMMANDÉ d'interdire aux services de les utiliser.

3.4.1 Message Motd

Commande : MOTD
Paramètres : [<cible>]

La commande MOTD est utilisée pour obtenir le "Mot d'ordre du jour" d'un certain serveur, ou du serveur actuel si <cible> est omis.

Les caractères génériques sont autorisés dans le paramètre <cible>.

Réponses numériques :

RPL_MOTDSTART	RPL_MOTD
RPL_ENDOFMOTD	ERR_NOMOTD

3.4.2 Message Lusers

Commande : LUSERS
Paramètres : [<gabarit> [<cible>]]

La commande LUSERS est utilisée pour obtenir des statistiques sur la taille du réseau IRC. Si aucun paramètre n'est donné, la réponse sera sur l'ensemble du réseau. Si un <gabarit> est spécifié, la réponse ne concernera alors que la partie du réseau formée par les serveurs qui correspondent au gabarit. Enfin, si le paramètre <cible> est spécifié, la demande sera transmise au serveur qui va générer la réponse.

Les caractères génériques sont autorisés dans le paramètre <cible>.

Réponses numériques :

RPL_LUSERCLIENT	RPL_LUSEROP
RPL_LUSERUNKOWN	RPL_LUSERCHANNELS
RPL_LUSERME	ERR_NOSUCHSERVER

3.4.3 Message Version

Commande : VERSION
Paramètres : [<cible>]

La commande VERSION est utilisée pour demander la version du programme de serveur. Un paramètre facultatif <cible> est utilisé pour demander la version du programme de serveur auquel un client n'est pas directement connecté.

Les caractères génériques sont autorisés dans le paramètre <cible>.

Réponses numériques :

ERR_NOSUCHSERVER	RPL_VERSION

Exemple :

VERSION tolsun.oulu.fi	; Commande pour vérifier la version du serveur "tolsun.oulu.fi".

3.4.4 Message Stats

Commande : STATS
Paramètres : [<interrogation> [<cible>]]

La commande stats est utilisée pour demander les statistiques d'un certain serveur. Si le paramètre <interrogation> est omis, seule la fin de la réponse stats est renvoyée.

Une interrogation peut être formulée pour toute lettre seule qui est seulement vérifiée par le serveur de destination et est autrement passée aux serveurs intermédiaires, ignorée et non altérée.

Les caractères génériques sont autorisés dans le paramètre <cible>.

Sauf pour ceux qui figurent ci-dessous, la liste des interrogations valides dépend de la mise en œuvre. Les interrogations standard figurant ci-dessous DEVRAIENT être prises en charge par le serveur :
l – retourne une liste des connexions du serveur, montrant depuis combien de temps chaque connexion a été établie et le trafic sur cette connexion en koctets et les messages pour chaque direction ;
m – retourne le compte d'usage pour chaque commande prise en charge par le serveur ; les commandes pour lesquelles le compte d'usage est zéro PEUVENT être omises ;
o – retourne une liste des utilisateurs privilégiés et des opérateurs configurés ;
u – retourne une chaîne montrant depuis combien de temps le serveur est en fonction.

Il est aussi RECOMMANDÉ que la configuration d'accès du client et du serveur soit publiée de cette façon.

Réponses numériques :
ERR_NOSUCHSERVER
RPL_STATSLINKINFO RPL_STATSUPTIME
RPL_STATSCOMMANDS RPL_STATSOLINE
RPL_ENDOFSTATS

Exemple :
STATS m ; Commande pour vérifier l'usage de la commande pour le serveur auquel on est connecté.

3.4.5 Message Links

Commande : LINKS
Paramètres : [[<serveur distant>] <gabarit de serveur>]

Avec LINKS, un utilisateur peut faire la liste de tous les noms de serveur qui sont connus du serveur qui répond à l'interrogation. La liste de serveurs retournée DOIT correspondre au gabarit, ou si un gabarit n'est pas donné, la liste complète est retournée.

Si <serveur distant> est donné en plus de <gabarit de serveur>, la commande LINKS est transmise au premier serveur trouvé qui correspond à ce nom (s'il en est) et ce serveur est alors obligé de répondre à l'interrogation.

Réponses numériques :
ERR_NOSUCHSERVER
RPL_LINKS RPL_ENDOFLINKS

Exemples :
LINKS *.au ; Commande pour faire la liste de tous les serveurs qui ont un nom correspondant à *.au;
LINKS *.edu *.bu.edu ; Commande pour faire la liste des serveurs correspondant à *.bu.edu telle que vue par le premier serveur correspondant à *.edu.

3.4.6 Message Time

Commande : TIME
Paramètres : [<cible>]

La commande time est utilisée pour demander l'heure locale au serveur spécifié. Si le paramètre <cible> n'est pas fourni, le serveur qui reçoit la commande doit répondre à l'interrogation.

Les caractères génériques sont autorisés dans le paramètre <cible>.

Réponses numériques :
ERR_NOSUCHSERVER RPL_TIME

Exemple :
TIME tolsun.oulu.fi ; vérifie l'heure sur le serveur "tolson.oulu.fi"

3.4.7 Message Connect

Commande : CONNECT
Paramètres : <serveur cible> <accès> [<serveur distant>]

La commande CONNECT peut être utilisée pour demander à un serveur d'essayer d'établir immédiatement une nouvelle connexion avec un autre serveur. CONNECT est une commande privilégiée et DEVRAIT n'être disponible qu'aux opérateurs IRC. Si un <serveur distant> est donné et si son gabarit ne correspond pas au nom du serveur qui analyse, la tentative de CONNECT est envoyée à la première correspondance de serveur distant. Autrement, la tentative de CONNECT est faite par le serveur qui traite la demande.

Le serveur qui reçoit une commande CONNECT distante DEVRAIT générer un message WALLOPS décrivant la source et la cible de la demande.

Réponses numériques :
ERR_NOSUCHSERVER ERR_NOPRIVILEGES
ERR_NEEDMOREPARAMS

Exemple :
CONNECT tolsun.oulu.fi 6667 ; Commande pour tenter de connecter le serveur local à tolsun.oulu.fi sur l'accès 6667.

3.4.8 Message Trace

Commande : TRACE
Paramètres : [<cible>]

La commande TRACE est utilisée pour trouver le chemin vers un serveur spécifique et des informations sur ses homologues. Chaque serveur qui traite cette commande DOIT en faire rapport à son envoyeur. Les réponses provenant des liaisons traversées forment une chaîne, qui montre le chemin vers la destination. Après avoir renvoyé cette réponse, l'interrogation DOIT être envoyée au prochain serveur jusqu'à ce que soit atteint le serveur <cible> désigné.

La commande TRACE est utilisée pour trouver le chemin vers un serveur spécifique. Chaque serveur qui traite ce message DOIT le dire à l'envoyeur par l'envoi d'une réponse qui indique qu'il est une liaison traversée, formant une chaîne de réponses. Après le renvoi de cette réponse, il DOIT alors envoyer le message TRACE au prochain serveur jusqu'à ce que le serveur spécifié soit atteint. Si le paramètre <cible> est omis, il est RECOMMANDÉ que la commande TRACE envoie un message à l'envoyeur pour lui dire à quels serveurs le serveur local a une connexion directe.

Si la destination donnée par <cible> est un serveur réel, il est EXIGÉ du serveur de destination qu'il fasse rapport de tous les serveurs, services et opérateurs qui lui sont connectés ; si la commande a été produite par un opérateur, le serveur PEUT aussi faire rapport de tous les utilisateurs qui lui sont connecté. Si la destination donnée par <cible> est un pseudonyme, seule une réponse pour ce pseudonyme est alors donnée. Si le paramètre <cible> est omis, il est RECOMMANDÉ que la commande TRACE soit analysée comme ciblée sur le serveur de traitement.

Les caractères génériques sont autorisés dans le paramètre <cible> .

Réponses numériques :
ERR_NOSUCHSERVER

Si le message TRACE est destiné à un autre serveur, tous les serveurs intermédiaires doivent retourner une réponse RPL_TRACELINK pour indiquer que TRACE est passé à travers eux et où il allait ensuite.

RPL_TRACELINK

Une réponse TRACE peut être composée d'un nombre quelconque des réponses numériques suivantes.
RPL_TRACECONNECTING RPL_TRACEHANDSHAKE
RPL_TRACEUNKNOWN RPL_TRACEOPERATOR
RPL_TRACEUSER RPL_TRACESERVER
RPL_TRACESERVICE RPL_TRACENEWTYPE
RPL_TRACECLASS RPL_TRACELOG
RPL_TRACEEND

Exemple :
TRACE *.oulu.fi ; TRACE vers un serveur correspondant à *.oulu.fi

3.4.9 Commande Admin

Commande : ADMIN
Paramètres : [<cible>]

La commande admin est utilisée pour trouver des informations sur l'administrateur du serveur désigné, ou du serveur actuel si le paramètre <cible> est omis. Chaque serveur DOIT avoir la capacité de transmettre les messages ADMIN aux autres serveurs.

Les caractères génériques sont autorisés dans le paramètre <cible>.

Réponses numériques :
ERR_NOSUCHSERVER
RPL_ADMINME RPL_ADMINLOC1
RPL_ADMINLOC2 RPL_ADMINEMAIL

Exemples :
ADMIN tolsun.oulu.fi ; demande une réponse ADMIN de la part de tolsun.oulu.fi
ADMIN syrk ; demande ADMIN pour le serveur auquel l'utilisateur syrk est connecté.

3.4.10 Commande Info

Commande : INFO
Paramètres : [<cible>]

La commande INFO est EXIGÉE pour retourner des informations qui décrivent le serveur : sa version, quand il a été compilé, le niveau de correction, quand il a été démarré, et toutes les autres informations diverses qui peuvent être considérées comme pertinentes.

Les caractères génériques sont autorisés dans le paramètre <cible>.

Réponses numériques :
ERR_NOSUCHSERVER
RPL_INFO RPL_ENDOFINFO

Exemples :
INFO csd.bu.edu ; demande une réponse INFO de la part de csd.bu.edu
INFO Angel ; demande des informations de la part du serveur auquel Angel est connecté.

3.5 Interrogations et commandes de service

Le groupe de commandes d'interrogation de service a été conçu pour retourner des informations sur tout service qui est connecté au réseau.

3.5.1 Message Servlist

Commande : SERVLIST
Paramètres : [<gabarit> [<type>]]

La commande SERVLIST est utilisée pour faire la liste des services actuellement connectés au réseau et visibles à l'utilisateur qui produit la commande. Les paramètres facultatifs peuvent être utilisés pour restreindre le résultat de l'interrogation (aux noms de services qui correspondent, et aux types de service).

Réponses numériques :
RPL_SERVLIST RPL_SERVLISTEND

3.5.2 Squery

Commande : SQUERY
Paramètres : <nom de service> <texte>

La commande SQUERY est utilisée de la même façon que PRIVMSG. La seule différence est que le receveur DOIT être un service. C'est la seule façon qu'a un message de texte d'être livré à un service.

Voir à PRIVMSG *(paragraphe 3.3.1)* des précisions sur les réponses et des exemples.

Exemples :
SQUERY irchelp :HELP privmsg Message au service avec le pseudonyme irchelp.
SQUERY dict@irc.fr :fr2en blaireau Message au service avec le nom dict@irc.fr.

3.6 Interrogations fondées sur l'utilisateur

Les interrogations d'utilisateur sont un groupe de commandes qui concernent principalement la découverte de précisions sur un utilisateur ou groupe d'utilisateurs particulier. Lorsque on utilise des caractères génériques avec l'une de ces commandes, si elles correspondent, elles vont retourner des informations seulement sur les utilisateurs qui sont 'visibles'. La visibilité d'un utilisateur est déterminée comme une combinaison du mode de l'utilisateur et de l'ensemble commun de canaux sur lesquels sont à la fois l'utilisateur et l'interrogateur.

Bien que les services NE DEVRAIENT PAS utiliser cette classe de messages, ils en ont l'autorisation.

3.6.1 Interrogation Who

Commande : WHO
Paramètres : [<gabarit> ["o"]]

La commande WHO est utilisée par un client pour générer une interrogation qui retourne une liste d'informations qui 'correspondent' au paramètre <gabarit> donné par le client. En l'absence du paramètre <gabarit>, elle fait la liste de tous les utilisateurs visibles (utilisateurs qui ne sont pas invisibles (mode d'utilisateur +i) et qui n'ont pas de canaux communs avec le client demandeur). On peut obtenir le même résultat en utilisant un <gabarit> de "0" ou de tout caractère générique qui va finir par correspondre à chaque utilisateur visible.

Le <gabarit> passé à WHO est confronté à l'hôte, serveur nom réel et pseudonyme des utilisateurs si les canaux <gabarit> ne peuvent pas être trouvés.

Si le paramètre "o" est passé, seuls les opérateurs sont retournés, conformément au <gabarit> fourni.

Réponses numériques :
ERR_NOSUCHSERVER
RPL_WHOREPLY RPL_ENDOFWHO

Exemples :
WHO *.fi ; Commande pour faire la liste de tous les utilisateurs qui correspondent pour "*.fi".
WHO jto* o ; Commande pour faire la liste de tous les utilisateurs qui ont une correspondance avec "jto*" si ils sont opérateurs.

3.6.2 Interrogation Whois

Commande : WHOIS
Paramètres : [<cible>] <gabarit> *("," <gabarit>)

Cette commande est utilisée pour demander des informations sur un utilisateur particulier. Le serveur va répondre à cette commande avec plusieurs messages numériques qui indiquent les différents statuts de chaque utilisateur qui correspond au gabarit (si le demandeur a le droit de les voir). Si aucun caractère générique n'est présent dans le <gabarit>, toute information autorisée au demandeur sur ce pseudonyme sera présentée.

Si le paramètre <cible> est spécifié, l'interrogation est envoyée à un serveur spécifique. Elle est utile si on veut savoir depuis combien de temps l'utilisateur en question a été inactif, car seul le serveur local (c'est-à-dire, le serveur auquel l'utilisateur est directement connecté) connaît ces informations, alors que tout le reste est connu de tous.

Les caractères génériques sont autorisés dans le paramètre <cible>.

Réponses numériques :

ERR_NOSUCHSERVER	ERR_NONICKNAMEGIVEN
RPL_WHOISUSER	RPL_WHOISCHANNELS
RPL_WHOISCHANNELS	RPL_WHOISSERVER
RPL_AWAY	RPL_WHOISOPERATOR
RPL_WHOISIDLE	ERR_NOSUCHNICK
RPL_ENDOFWHOIS	

Exemples :

WHOIS wiz	; retourne les informations d'utilisateur disponibles sur le pseudonyme WiZ.
WHOIS eff.org trillian	; demande au serveur eff.org les informations d'utilisateur sur trillian.

3.6.3 Interrogation Whowas

Commande : WHOWAS
Paramètres : <pseudonyme> *("," <pseudonyme>) [<compte> [<cible>]]

Whowas demande des informations sur un pseudonyme qui n'existe plus. Cela peut être dû à un changement de pseudonyme où au départ de l'utilisateur de l'IRC. En réponse à cette interrogation, le serveur cherche dans son historique des pseudonymes, examinant tous les alias qui sont lexicalement les mêmes (pas de correspondance avec caractère générique ici). L'historique est remonté, pour retourner d'abord les entrées les plus récentes. Si il y a plusieurs entrées, jusqu'à <compte> réponses seront retournées (ou toutes si aucun paramètre <compte> n'est donné). Si un nombre non positif est passé pour <compte>, une recherche complète est alors effectuée.

Les caractères génériques sont autorisés dans le paramètre <cible>.

Réponses numériques :

ERR_NONICKNAMEGIVEN	ERR_WASNOSUCHNICK
RPL_WHOWASUSER	RPL_WHOISSERVER
RPL_ENDOFWHOWAS	

Exemples :

WHOWAS Wiz	; retourne toutes les information de l'historique des pseudos sur le pseudonyme "WiZ";
WHOWAS Mermaid 9	; retourne au plus, les neuf plus récentes entrées de l'historique des pseudonymes sur "Mermaid";
WHOWAS Trillian 1 *.edu	; retourne l'historique le plus récent pour "Trillian" à partir du premier serveur trouvé pour correspondre à "*.edu".

3.7 Messages divers

Les messages de cette catégorie ne rentrent dans aucune des catégories précédentes mais font néanmoins partie du protocole et sont EXIGÉS.

3.7.1 Message Kill

Commande : KILL
Paramètres : <pseudonyme> <commentaire>

La commande KILL est utilisée pour causer la clôture d'une connexion client-serveur par le serveur qui a cette connexion. Les serveurs génèrent des messages KILL en cas de collision de pseudonyme. Elle PEUT aussi être disponible aux utilisateurs qui ont le statut d'opérateur.

Les clients qui ont des algorithmes de reconnexion automatique rendent effectivement cette commande inutile car la déconnexion n'est que brève. Elle casse cependant le flux des données et peut être utilisée pour arrêter de grosses quantités de données en 'arrosage' de la part d'utilisateurs abusifs ou par accident. Les utilisateurs abusifs ne s'en soucient d'habitude guère car ils vont se reconnecter promptement et reprendre leur comportement abusif. Pour empêcher l'abus de cette commande, tout utilisateur peut choisir de recevoir les messages KILL générés pour d'autres pour 'garder un œil' sur les trublions potentiels.

Dans un domaine où il est EXIGÉ que les pseudonymes soient uniques au monde en tout temps, les messages KILL sont envoyés chaque fois que des 'dupliqués' sont détectés (c'est une tentative pour enregistrer deux utilisateurs avec le même pseudonyme) dans l'espoir que les deux vont disparaître et qu'un seul réapparaîtra.

Lorsque un client est retiré par suite d'un message KILL, le serveur DEVRAIT ajouter le pseudonyme à la liste des pseudonymes indisponibles pour tenter d'éviter que des clients réutilisent immédiatement ce nom ce qui est généralement un schéma de comportement abusif menant souvent à des "boucles de KILL" inutiles. Voir dans la [RFC2813] "Protocole du serveur IRC" plus d'informations sur cette procédure.

Le commentaire donné DOIT refléter la raison réelle du KILL. Pour les KILL générés par le serveur, c'est généralement constitué de détails concernant l'origine des deux pseudonymes en conflit. Pour ceux des utilisateurs, il leur appartient de fournir une raison adéquate pour satisfaire ceux qui la voient. Pour empêcher/décourager la génération de KILL frauduleux pour cacher l'identité de l'envoyeur de KILL, le commentaire montre aussi un 'chemin de kill' qui est mis à jour par chaque serveur au travers duquel il passe, chacun ajoutant son nom au chemin.

Réponses numériques :
ERR_NOPRIVILEGES ERR_NEEDMOREPARAMS
ERR_NOSUCHNICK ERR_CANTKILLSERVER

Note : Il est RECOMMANDÉ que seuls les opérateurs soient autorisés à éliminer d'autres utilisateurs avec la commande KILL. Cette commande a été l'objet de nombreuses controverses pendant des années, et lors de la rédaction de la présente recommandation, il a été aussi largement reconnu que même les opérateurs ne devraient pas être autorisés à éliminer des utilisateurs sur des serveurs distants.

3.7.2 Message Ping

Commande : PING
Paramètres : <serveur1> [<serveur2>]

La commande PING est utilisée pour vérifier la présence d'un client ou serveur actif à l'autre bout de la connexion. Les serveurs envoient un message PING à intervalles réguliers si aucune autre activité n'est détectée en provenance d'une connexion. Si une connexion ne répond pas à un message PING dans un certain délai, cette connexion est close. Un message PING PEUT être envoyé même si la connexion est active.

Lorsque un message PING est reçu, le message PONG approprié DOIT être envoyé comme réponse au <serveur1> (le serveur qui a envoyé le message PING) aussitôt que possible. Si le paramètre <serveur2> est spécifié, il représente la cible du ping, et le message lui est transmis.

Réponses numériques :
ERR_NOORIGIN ERR_NOSUCHSERVER

Exemples :
PING tolsun.oulu.fi ; Commande d'envoi d'un message PING au serveur.
PING WiZ tolsun.oulu.fi ; Commande de WiZ d'envoi d'un message PING au serveur "tolsun.oulu.fi".
PING :irc.funet.fi ; Message Ping envoyé par le serveur "irc.funet.fi".

3.7.3 Message Pong

Commande : PONG
Paramètres : <serveur> [<serveur2>]

Le message PONG est une réponse au message Ping. Si le paramètre <serveur2> est donné, ce message DOIT être transmis à la cible désignée. Le paramètre <serveur> est le nom de l'entité qui a répondu au message PING et généré ce message.

Réponses numériques :
ERR_NOORIGIN ERR_NOSUCHSERVER

Exemple :
PONG csd.bu.edu tolsun.oulu.fi ; Message PONG provenant de csd.bu.edu à tolsun.oulu.fi

3.7.4 Erreur

Commande : ERROR
Paramètres : <message d'erreur>

La commande ERROR est à utiliser par les serveurs lorsque ils font rapport d'une erreur sérieuse ou fatale à leurs homologues. Elle peut aussi être envoyée d'un serveur à un autre mais NE DOIT être acceptée d'aucun client normal inconnu.

Seul un message ERROR DEVRAIT être utilisé pour rapporter des erreurs qui surviennent sur une liaison de serveur à serveur. Un message ERROR est envoyé au serveur à l'autre extrémité (qui en fait rapport aux utilisateurs et journaux d'événements locaux appropriés) et aux utilisateurs et journaux d'événements locaux appropriés. Il n'est pas à passer par un serveur à d'autres serveurs si il est reçu d'un serveur.

Le message ERROR est aussi utilisé avant de mettre un terme à une connexion de client.

Lorsque un serveur envoie à ses opérateurs un message ERROR reçu, le message DEVRAIT être encapsulé dans un message NOTICE, ce qui indique que le client n'était pas responsable de l'erreur.

Réponse numérique : aucune.

Exemples :
ERROR :Server *.fi existe déjà ; message ERROR à l'autre serveur qui a causé cette erreur.
NOTICE WiZ :ERROR de csd.bu.edu -- Serveur *.fi existe déjà ; même message ERROR que ci-dessus mais envoyé à l'utilisateur WiZ sur l'autre serveur.

4. Caractéristiques facultatives

La présente section décrit des messages FACULTATIFS. Ils ne sont pas obligés dans une mise en œuvre de serveur conforme au protocole décrit ici. En l'absence de la caractéristique, un message de réponse d'erreur ou une erreur Commande inconnue DOIT être généré. Si le message est destiné à un autre serveur pour répondre, il DOIT alors être transmis (une analyse élémentaire est EXIGÉE). Les valeurs numériques pour cela sont données ci-dessous avec les messages.

Dans cette section, seuls les messages USERHOST et ISON sont disponibles aux services.

4.1 Away

Commande : AWAY
Paramètres : [<texte>]

Avec une commande AWAY, les clients peuvent établir une chaîne de réponse automatique pour toute commande PRIVMSG dirigée sur eux (pas sur les canaux sur lesquels ils sont). Le serveur envoie une réponse automatique au client qui envoie la commande PRIVMSG. Le seul serveur qui répond est celui auquel est connecté le client envoyeur.

La commande AWAY est utilisée soit avec un paramètre, pour établir un message AWAY, soit sans paramètre, pour retirer le message AWAY.

À cause de son coût élevé (en mémoire et en bande passante) le message AWAY DEVRAIT n'être utilisé que pour la communication client-serveur. Un serveur PEUT choisir d'ignorer en silence les messages AWAY reçus des autres serveurs. Pour mettre à jour l'état 'away' d'un client parmi des serveurs, le mode d'utilisateur 'a' DEVRAIT plutôt être utilisé (voir au paragraphe 3.1.5).

Réponses numériques :
RPL_UNAWAY RPL_NOWAWAY

Exemple :
AWAY :Parti déjeuner. Retour dans 5 ; Commande pour régler le message away à "Parti déjeuner. Retour dans 5".

4.2 Message Rehash

Commande : REHASH
Paramètres : None

La commande rehash est une commande administrative qui peut être utilisée par un opérateur pour forcer le serveur à relire

et retraiter son fichier de configuration.

Réponses numériques :
RPL_REHASHING ERR_NOPRIVILEGES

Exemple :
REHASH ; message d'utilisateur avec statut d'opérateur demandant au serveur de relire son fichier de configuration.

4.3 Message Die

Commande : DIE
Paramètres : Aucun

Un opérateur peut utiliser la commande DIE pour fermer le serveur. Ce message est facultatif car il peut être vu comme un risque qu'ell permette à des personnes quelconques de se connecter comme opérateur sur un serveur et d'exécuter cette commande.

La commande DIE DOIT toujours être entièrement traitée par le serveur auquel est connecté le client envoyeur et NE DOIT PAS être passée aux autres serveurs connectés.

Réponse numérique : ERR_NOPRIVILEGES

Exemple :
DIE ; aucun paramètre n'est exigé.

4.4 Message Restart

Commande : RESTART
Paramètres : Aucun

Un opérateur peut utiliser la commande restart pour forcer le serveur à redémarrer. Ce message est facultatif car il peut être vu comme le risque de permettre à des personnes arbitraires de se connecter comme un opérateur à un serveur et d'exécuter cette commande, causant (au moins) une interruption de service.

La commande RESTART DOIT toujours être entièrement traités par le serveur auquel est connecté le client envoyeur et NE DOIT PAS être passée aux autres serveurs connectés.

Réponses numériques : ERR_NOPRIVILEGES

Exemple :
RESTART ; aucun paramètre n'est exigé.

4.5 Message Summon

Commande : SUMMON
Paramètres : <utilisateur> [<cible> [<canal>]]

La commande SUMMON *(convoquer)* peut être utilisée pour donner aux utilisateurs qui sont sur un hôte qui fonctionne avec un serveur IRC un message leur demandant de se joindre à IRC. Ce message n'est envoyé que si le serveur cible (a) a la capacité SUMMON, (b) si l'utilisateur est connecté, et (c) si le processus de serveur peut écrire au tty (ou similaire) de l'utilisateur.

Si aucun paramètre<serveur> n'est donné, il essaye de convoquer l'<utilisateur> à partir du serveur auquel le client est connecté qui est supposé être la cible.

Si summon n'est pas activé dans un serveur, il DOIT retourner le numéro ERR_SUMMONDISABLED.

Réponses numériques :
ERR_NORECIPIENT ERR_FILEERROR
ERR_NOLOGIN ERR_NOSUCHSERVER
ERR_SUMMONDISABLED RPL_SUMMONING

Exemples :
SUMMON jto ; convoque l'utilisateur jto sur l'hôte du serveur.
SUMMON jto tolsun.oulu.fi ; convoque l'utilisateur jto sur l'hôte sur lequel fonctionne un serveur nommé "tolsun.oulu.fi".

4.6 Commande Users

Commande : USERS
Paramètres : [<cible>]

La commande USERS retourne une liste d'utilisateurs enregistrés auprès du serveur sous un format similaire aux commandes UNIX who(1), rusers(1) et finger(1). Si elle est désactivée, le numéro correct DOIT être retourné pour l'indiquer.

À cause des implications d'une telle commande pour la sécurité, elle DEVRAIT être désactivée par défaut dans les mises en œuvre de serveur. L'activer DEVRAIT exiger de recompiler le serveur ou quelque changement équivalent plutôt que de simplement basculer une option et redémarrer le serveur. La procédure pour activer cette commande DEVRAIT aussi inclure des longs commentaires appropriés.

Réponses numériques :
ERR_NOSUCHSERVER ERR_FILEERROR
RPL_USERSSTART RPL_USERS
RPL_NOUSERS RPL_ENDOFUSERS
ERR_USERSDISABLED

Réponse désactivée : ERR_USERSDISABLED

Exemple :
USERS eff.org ; demande une liste des utilisateurs enregistrés auprès du serveur eff.org

4.7 Message Operwall

Commande : WALLOPS
Paramètres : <texte à envoyer>

La commande WALLOPS est utilisée pour envoyer un message à tous les utilisateurs actuellement connectés qui ont réglé le mode d'utilisateur 'w' pour eux-mêmes (voir au paragraphe 3.1.5 "Modes d'utilisateur").

Après avoir mis en œuvre WALLOPS comme commande d'utilisateur, il a été trouvé qu'il en était souvent fait un usage abusif comme moyen d'envoyer un message à un grand nombre de gens. À cause de cela, il est RECOMMANDÉ que la mise en œuvre de WALLOPS ne permette et reconnaisse que les serveurs comme générateurs de WALLOPS.

Réponse numérique : ERR_NEEDMOREPARAMS

Exemple :
:csd.bu.edu WALLOPS :Connect '*.uiuc.edu 6667' de Joshua ; message WALLOPS de csd.bu.edu annonçant un message CONNECT reçu de Joshua et traité.

4.8 Message Userhost

Commande : USERHOST
Paramètres : <pseudonyme> *(ESPACE <pseudonyme>)

La commande USERHOST prend une liste de jusqu'à cinq pseudonymes, chacun séparé par un caractère espace, et retourne une liste d'informations sur chaque pseudonyme trouvé. La liste retournée a chaque réponse séparée par une espace.

Réponses numériques :
RPL_USERHOST ERR_NEEDMOREPARAMS

Exemple :
USERHOST Wiz Michael syrk ; demande USERHOST d'informations sur les pseudonymes "Wiz",

"Michael", et "syrk"
:ircd.stealth.net 302 yournick :syrk=+syrk@millennium.stealth.net ; Réponse pour l'utilisateur syrk.

4.9 Message Ison

Commande : ISON
Paramètres : <pseudonyme> *(ESPACE <pseudonyme>)

La commande ISON a été mise en œuvre pour fournir un moyen rapide et efficace pour obtenir une réponse à la question de savoir si un certain pseudonyme est actuellement sur IRC. ISON ne prend qu'un (1) type de paramètre : une liste de pseudonymes séparés par une espace. Pour chaque pseudonyme présent dans la liste, le serveur l'ajoute à sa chaîne de réponse. Donc, la chaîne de réponse peut retourner une chaîne vide (aucun des pseudonymes cités n'est présent) une copie exacte de la chaîne de paramètres (tous sont présents) ou tout autre sous-ensemble de l'assortiment de pseudonymes donnés dans le paramètre. La seule limite au nombre de pseudonymes qui peuvent être vérifiés est que la longueur combinée NE DOIT PAS être si grande qu'elle oblige le serveur à la découper afin qu'elle tienne en 512 caractères.

ISON n'est traité que par le serveur local pour le client qui envoie la commande et n'est donc pas passé aux autres serveurs pour d'autres traitements.

Réponses numériques :
RPL_ISON ERR_NEEDMOREPARAMS

Exemple :
ISON phone trillMessage ian WiZ jarlek Avalon Angel Monstah syrk ; Exemple de demande ISON pour 7 pseudonymes.

5. Réponses

Voici la liste des réponses numériques qui sont générées en réponse aux commandes données précédemment. Chaque réponse numérique est donnée avec son numéro, son nom et sa chaîne de réponse.

5.1 Réponses aux commandes

Les numéros dans la gamme de 001 à 099 sont utilisés seulement pour la connexion client-serveur et ne devraient jamais voyager entre les serveurs. Les réponses générées dans la réponse aux commandes se trouvent dans la gamme de 200 à 399.

001 RPL_WELCOME "Bienvenue sur le relais de causette Internet <pseudonyme>!<utilisateur>@<hôte>"
002 RPL_YOURHOST "Votre hôte est <nom de serveur>, version en cours <ver>"
003 RPL_CREATED "Ce serveur a été créé <date>"
004 RPL_MYINFO "<nom de serveur> <version> <mode d'utilisateur disponible> <mode de canal disponible>"
- Le serveur envoie les réponses 001 à 004 à un utilisateur lors d'un enregistrement réussi.

005 RPL_BOUNCE "Essayer le serveur <nom de serveur>, accès <numéro d'accès>"
- Envoyé par le serveur à un utilisateur pour suggérer un serveur de remplacement. C'est souvent utilisé lorsque la connexion est refusée parce que le serveur est déjà plein.

302 RPL_USERHOST ":*1<réponse> *(" " <réponse>)"
- Format de réponse utilisé par USERHOST pour énumérer les réponses à la liste d'interrogations. La chaîne de réponse est composée comme suit :
  réponse = pseudonyme ["*"] "=" ("+" / "-") nom d'hôte
  Le '*' indique si le client est enregistré comme opérateur. Le caractère '-' ou '+' représente si le client établit un message AWAY ou non.

303 RPL_ISON ":*1<pseudonyme> *(" " <pseudonyme>)"
- Format de réponse utilisé par ISON pour faire la liste des réponses à la liste d'interrogations.

301 RPL_AWAY "<pseudonyme> :<message away>"
305 RPL_UNAWAY ":Vous n'êtes plus marqué comme étant parti"
306 RPL_NOWAWAY ":Vous avez été marqué comme étant parti"
- Ces réponses sont utilisées avec la commande AWAY (si elle est permise). RPL_AWAY est envoyé à tout client qui envoie un PRIVMSG à un client qui est parti. RPL_AWAY n'est envoyé que par le serveur auquel le client est

connecté. Les réponses RPL_UNAWAY et RPL_NOWAWAY sont envoyées lorsque le client retire et crée un message AWAY.

311 RPL_WHOISUSER "<pseudonyme> <utilisateur> <hôte> * :<nom réel>"
312 RPL_WHOISSERVER "<pseudonyme> <serveur> :<info de serveur>"
313 RPL_WHOISOPERATOR "<pseudonyme> :est un opérateur IRC"
317 RPL_WHOISIDLE "<pseudonyme> <entier> :secondes de repos"
318 RPL_ENDOFWHOIS "<pseudonyme> :Fin de liste WHOIS"
319 RPL_WHOISCHANNELS "<pseudonyme> :*(("@" / "+") <canal> " ")"

- Les réponses 311 - 313, 317 - 319 sont toutes des réponses générées en réponse à un message WHOIS. Étant donné qu'il y a assez de paramètres présents, le serveur qui répond DOIT soit formuler une réponse parmi les numéros ci-dessus (si le pseudonyme objet de l'interrogation est trouvé) soit retourner une réponse d'erreur. Le '*' dans RPL_WHOISUSER est là comme caractère littéral et non comme caractère générique. Pour chaque réponse donnée, seul RPL_WHOISCHANNELS peut apparaître plus d'une fois (pour de longues listes de noms de canal). Les caractères '@' et '+' après le nom de canal indiquent si un client est un opérateur de canal ou si il a reçu la permission de parler sur un canal à modérateur. La réponse RPL_ENDOFWHOIS est utilisée pour marquer la fin du traitement d'un message WHOIS.

314 RPL_WHOWASUSER "<pseudonyme> <utilisateur> <hôte> * :<nom réel>"
369 RPL_ENDOFWHOWAS "<pseudonyme> :Fin de WHOWAS"

- Quand il répond à un message WHOWAS, un serveur DOIT utiliser les réponses RPL_WHOWASUSER, RPL_WHOISSERVER ou ERR_WASNOSUCHNICK pour chaque pseudonyme de la liste présentée. À la fin de tous les lots de réponses, il doit y avoir RPL_ENDOFWHOWAS (même si il n'y a qu'une seule réponse et qu'elle est erronée).

321 RPL_LISTSTART Obsolète. Non utilisé.
322 RPL_LIST "<canal> <# visible> :<sujet>"
323 RPL_LISTEND ":Fin de LIST"

- Les réponses RPL_LIST, RPL_LISTEND marquent les réponses réelles avec des données et terminent la réponse du serveur à une commande LIST. Si il n'y a pas de canal disponible pour le retour, seule la fin de la réponse DOIT être envoyée.

325 RPL_UNIQOPIS "<canal> <pseudonyme>"
324 RPL_CHANNELMODEIS "<canal> <mode> <mode params>"
331 RPL_NOTOPIC "<canal> :Aucun sujet n'est établi"
332 RPL_TOPIC "<canal> :<sujet>"

- Lors de l'envoi d'un message TOPIC pour déterminer le sujet d'un canal, une des deux réponses est envoyée. Si le sujet est établi, RPL_TOPIC est renvoyé, autrement, c'est RPL_NOTOPIC.

341 RPL_INVITING "<pseudonyme> <canal>"

- Retourné par le serveur pour indiquer que le message INVITE tenté a réussi et est passé vers le client final.

342 RPL_SUMMONING "<utilisateur> :Convoque l'utilisateur sur IRC"

- Retourné par un serveur qui répond à un message SUMMON pour indiquer qu'il convoque cet utilisateur.

346 RPL_INVITELIST "<canal> <gabarit d'invite>"
347 RPL_ENDOFINVITELIST "<canal> :Fin de liste d'invite de canal"

- Lorsque il fait la liste des 'gabarits d'invitation' pour un certain canal, un serveur est obligé de renvoyer la liste en utilisant les messages RPL_INVITELIST et RPL_ENDOFINVITELIST. Un RPL_INVITELIST distinct est envoyé pour chaque gabarit actif. Après que la liste des gabarits a été faite (ou si il n'en est aucun de présent) un RPL_ENDOFINVITELIST DOIT être envoyé.

348 RPL_EXCEPTLIST "<canal> <gabarit d'exception>"
349 RPL_ENDOFEXCEPTLIST "<canal> :Fin de liste d'exception de canal"

- Lorsque il fait la liste des 'gabarits d'exception' pour un certain canal, un serveur est obligé de renvoyer la liste en utilisant les messages RPL_EXCEPTLIST et RPL_ENDOFEXCEPTLIST. Un RPL_EXCEPTLIST distinct est envoyé pour chaque gabarit actif. Après que la liste des gabarits a été établie (ou si il n'en est aucun de présent) un message RPL_ENDOFEXCEPTLIST DOIT être envoyé.

351 RPL_VERSION "<version>.<niveau débogage> <serveur> :<commentaires>"

- Réponse du serveur montrant les détails de sa version. La <version> est celle du logiciel utilisé (y compris toute révision de niveau de débogage) et le <niveau débogage> est utilisé pour indiquer si le serveur fonctionne en "mode débogage".

Le champ "commentaires" peut contenir des commentaires sur la version ou des précisions sur la version.

352 RPL_WHOREPLY "<canal> <utilisateur> <hôte> <serveur> <pseudonyme> ("H" / "G" > ["*"] [("@" / "+")] :<compte de bonds> <nom réel>"
315 RPL_ENDOFWHO "<nom> :Fin de liste WHO"

- La paire RPL_WHOREPLY et RPL_ENDOFWHO est utilisée pour répondre à un message WHO. Le message RPL_WHOREPLY n'est envoyé que si il y a une correspondance appropriée à l'interrogation WHO. Si il y a une liste de paramètres fournis avec un message WHO, une RPL_ENDOFWHO DOIT être envoyée après le traitement de chaque élément de la liste avec <nom> comme élément.

353 RPL_NAMREPLY "("=" / "*" / "@") <canal> :["@" / "+"] <pseudonyme> *(" " ["@" / "+"] <pseudonyme>)"

- "@" est utilisé pour les canaux secrets, "*" pour les canaux privés, et "=" pour les autres (canaux publics).

366 RPL_ENDOFNAMES "<canal> :Fin de la liste NAMES"

- Pour répondre à un message NAMES, une paire de réponses RPL_NAMREPLY et RPL_ENDOFNAMES est renvoyée par le serveur au client. Si il ne se trouve pas de canal comme dans l'interrogation, seul RPL_ENDOFNAMES est alors retourné. La seule exception à cela est lorsque un message NAMES est envoyé sans paramètre et que tous les canaux et contenus visibles sont renvoyés dans une série de messages RPL_NAMEREPLY avec RPL_ENDOFNAMES pour marquer la fin.

364 RPL_LINKS "<gabarit> <serveur> :<compte de bonds> <informations de serveur>"
365 RPL_ENDOFLINKS "<gabarit> :Fin de la liste LINKS"

- En répondant au message LINKS, un serveur DOIT renvoyer les réponses en utilisant le numéro RPL_LINKS et marquer la fin de la liste en utilisant une réponse RPL_ENDOFLINKS.

367 RPL_BANLIST "<canal> <gabarit d'interdiction>"
368 RPL_ENDOFBANLIST "<canal> :Fin de liste de gabarit d'interdiction de canal"

- Lorsque on fait la liste des "interdictions" actives pour un certain canal, un serveur est obligé de renvoyer la liste en utilisant les messages RPL_BANLIST et RPL_ENDOFBANLIST. Un RPL_BANLIST distinct est envoyé pour chaque gabarit d'interdiction actif. Après que la liste des gabarits d'interdiction a été faite (ou si il n'en est aucun de présent) un message RPL_ENDOFBANLIST DOIT être envoyé.

371 RPL_INFO ":<chaîne>"
374 RPL_ENDOFINFO ":Fin de liste INFO"

- Un serveur qui répond à un message INFO est obligé d'envoyer toutes ses 'info' dans une série de messages RPL_INFO avec une réponse RPL_ENDOFINFO pour indiquer la fin des réponses.

375 RPL_MOTDSTART ":- <serveur> Mot d'ordre du jour - "
372 RPL_MOTD ":- <texte>"
376 RPL_ENDOFMOTD ":Fin de la commande MOTD"

- Lors de la réponse au message MOTD quand le fichier MOTD est trouvé, le fichier est affiché ligne par ligne, chaque ligne ne dépassant pas 80 caractères, en utilisant le format RPL_MOTD des réponses. Celles-ci DOIVENT être entourées par un RPL_MOTDSTART (avant les RPL_MOTD) et un RPL_ENDOFMOTD (après).

381 RPL_YOUREOPER ":Vous êtes maintenant un opérateur IRC"

- RPL_YOUREOPER est renvoyé à un client qui vient de réussir à produire avec succès un message OPER et à obtenir le statut d'opérateur.

382 RPL_REHASHING "<fichier de configuration> :Recompilation"

- Si l'option REHASH est utilisée et si un opérateur envoie un message REHASH, un RPL_REHASHING est renvoyée à l'opérateur.

383 RPL_YORESERVICE "Vous êtes le service <nom du service>"

- Envoyé par le serveur à un service après réussite de l'enregistrement.

391 RPL_TIME "<serveur> :<chaîne indiquant l'heure locale du serveur>"

- En répondant au message TIME, un serveur DOIT envoyer la réponse en utilisant le format RPL_TIME ci-dessus. La chaîne qui donne l'heure doit seulement contenir le jour et l'heure corrects. Il n'y a pas d'autre exigence pour la chaîne d'horodatage.

392 RPL_USERSSTART ":Identifiant d'utilisateur Terminal Hôte"
393 RPL_USERS ":<nom d'utilisateur> <ligne tty> <nom d'hôte>"

394 RPL_ENDOFUSERS ":Fin d'utilisateurs"
395 RPL_NOUSERS ":Personne d'enregistré"

- Si le message USERS est traité par un serveur, les réponses RPL_USERSTART, RPL_USERS, RPL_ENDOFUSERS et RPL_NOUSERS sont utilisées. RPL_USERSSTART DOIT être envoyé en premier, suivi par une séquence de RPL_USERS ou d'un seul RPL_NOUSER. À la fin se trouve le RPL_ENDOFUSERS.

200 RPL_TRACELINK "Liaison <version & niveau débogage> <destination> <prochain serveur> V<version du protocole> <durée d'activité de la liaison en secondes> <sendq aval> <sendq amont>"
201 RPL_TRACECONNECTING "Essayer. <classe> <serveur>"
202 RPL_TRACEHANDSHAKE "H.S. <classe> <serveur>"
203 RPL_TRACEUNKNOWN "???? <classe> [<adresse IP du client en forme séparée par des points>]"
204 RPL_TRACEOPERATOR "Opérateur <classe> <pseudonyme>"
205 RPL_TRACEUSER "Utilisateur <classe> <pseudonyme>"
206 RPL_TRACESERVER "Serveur <classe> <int>S <int>C <serveur> <pseudonyme!utilisateur|*! *>@<hôte|serveur> V<protocole version>"
207 RPL_TRACESERVICE "Service <classe> <nom> <type> <type actif>"
208 RPL_TRACENEWTYPE "<nouveau type> 0 <nom du client>"
209 RPL_TRACECLASS "Classe <classe> <compte>"
210 RPL_TRACERECONNECT Non utilisé.
261 RPL_TRACELOG "Fichier <fichier de journalisation> <niveau de débogage>"
262 RPL_TRACEEND "<nom de serveur> <version & niveau de débogage> :Fin de TRACE"

- Les RPL_TRACE* sont toutes retournées par le serveur en réponse au message TRACE. Combien sont retournées dépend du message TRACE et de si elles sont envoyées par un opérateur ou non. Il n'y a pas d'ordre prédéfini pour celle qui vient en premier. Les réponses RPL_TRACEUNKNOWN, RPL_TRACECONNECTING et RPL_TRACEHANDSHAKE sont toutes utilisées pour des connexions qui n'ont pas été pleinement établies et sont inconnues, ou toujours en attente de connexion ou en cours d'achèvement de la "prise de contact de serveur". RPL_TRACELINK est renvoyé par tout serveur qui traite un message TRACE et doit le passer à un autre serveur. La liste des RPL_TRACELINK envoyés en réponse à une commande TRACE traversant le réseau IRC devrait refléter la connexité réelle des serveurs eux-mêmes le long de ce chemin.
  RPL_TRACENEWTYPE est à utiliser pour toute connexion qui ne rentre pas dans les autres catégories mais est quand même affichée. RPL_TRACEEND est envoyé pour indiquer la fin de la liste.

211 RPL_STATSLINKINFO "<nom de liaison> <sendq> <messages envoyés> <k octets envoyés> <messages reçus> <k octets reçus> <durée d'ouverture>"

- Rapporte les statistiques sur une connexion. <nom de liaison> identifie la connexion, <sendq> est la quantité de données en file d'attente pour être envoyées, <messages envoyés> le nombre de messages envoyés, et <k octets envoyés> la quantité de données envoyées, en k octets. <messages reçus> et <k octets reçus> sont l'équivalent de <messages envoyés> et <k octets reçus> pour les données reçues, respectivement, <durée d'ouverture> indique depuis combien de temps la connexion est ouverte, en secondes.

212 RPL_STATSCOMMANDS "<commande> <compte> <compte d'octets> <compte distant>"

- Rapporte les statistiques de l'utilisation des commandes.

219 RPL_ENDOFSTATS "<lettre statistique> :Fin de rapport de STATS"
242 RPL_STATSUPTIME ":Serveur actif %d jours %d:%02d:%02d"

- Rapports le temps d'activité du serveur.

243 RPL_STATSOLINE "O <gabarit d'hôte> * <nom>"

- Rapporte les hôtes admis à partir desquels l'utilisateur peut devenir opérateur IRC.

221 RPL_UMODEIS "<chaîne de mode d'utilisateur>"

- Pour répondre à une interrogation sur le mode du client, RPL_UMODEIS est renvoyé.

234 RPL_SERVLIST "<nom> <serveur> <gabarit> <type> <compte de bonds> <info>"
235 RPL_SERVLISTEND "<gabarit> <type> :Fin de liste de services"

- Lorsque il fait la liste des services en réponse à un message SERVLIST, un serveur est obligé de renvoyer la liste en utilisant les messages RPL_SERVLIST et RPL_SERVLISTEND. Un RPL_SERVLIST distinct est envoyé pour chaque service. Après que la liste des services a été faite (ou si aucun n'est présent) un RPL_SERVLISTEND DOIT être envoyé.

251 RPL_LUSERCLIENT ":Il y a <entier> utilisateurs et <entier> services sur <entier> serveurs"
252 RPL_LUSEROP "<entier> :opérateur(s) en ligne"
253 RPL_LUSERUNKNOWN "<entier> :connexion(s) inconnues"

254 RPL_LUSERCHANNELS "<entier> :canaux formés"
255 RPL_LUSERME ":J'ai <entier> clients et <entier> serveurs"
- En traitant un message LUSERS, le serveur envoie un ensemble de réponses à partir de RPL_LUSERCLIENT, RPL_LUSEROP, RPL_USERUNKNOWN, RPL_LUSERCHANNELS et RPL_LUSERME. Lorsque il répond, un serveur DOIT renvoyer RPL_LUSERCLIENT et RPL_LUSERME. Les autres réponses ne sont renvoyées que si un compte différent de zéro est trouvé pour elles.

256 RPL_ADMINME "<serveur> :Informations administratives"
257 RPL_ADMINLOC1 ":<Informations administratives>"
258 RPL_ADMINLOC2 ":<Informations administratives>"
259 RPL_ADMINEMAIL ":<Informations administratives>"
- Lorsque il répond à un message ADMIN, un serveur est supposé utiliser les réponses RPL_ADMINME à RPL_ADMINEMAIL et fournir un message textuel avec chacune d'elles. Pour RPL_ADMINLOC1 une description de la ville province et pays dans lequel se trouve le serveur est attendue, suivie par les détails de l'institution (RPL_ADMINLOC2) et finalement le contact administratif pour le serveur (une adresse de messagerie électronique est EXIGÉE ici) dans RPL_ADMINEMAIL.

263 RPL_TRYAGAIN "<commande> :Prière d'attendre un peu avant de recommencer."
- Lorsque un serveur abandonne une commande sans la traiter, il DOIT utiliser la réponse RPL_TRYAGAIN pour informer le client qui l'a générée.

5.2 Réponses d'erreur

Les réponses d'erreur se trouvent dans la gamme de 400 à 599.

401 ERR_NOSUCHNICK "<pseudonyme> :Pas de tel pseudonyme/canal"
- Utilisé pour indiquer que le paramètre de pseudonyme fourni pour une commande est actuellement non utilisé.

402 ERR_NOSUCHSERVER "<nom de serveur> :Pas de tel serveur"
- Utilisé pour indiquer que le nom de serveur donné n'existe pas actuellement.

403 ERR_NOSUCHCHANNEL ²"<nom de canal> :Pas de tel canal"
- Utilisé pour indiquer que le nom de canal donné est invalide.

404 ERR_CANNOTSENDTOCHAN "<nom de canal> :Pas possible d'envoyer au canal"
- Envoyé à un utilisateur qui n'est (a) pas sur un canal qui est en mode +n ou (b) pas un chanop (ou mode +v) sur un canal qui a le mode +m établi ou dont l'utilisateur est interdit et qui essaye d'envoyer un message PRIVMSG à ce canal.

405 ERR_TOOMANYCHANNELS "<nom de canal> :Vous vous êtes joint à trop de canaux"
- Envoyé à un utilisateur lorsque il s'est joint au nombre maximum admis de canaux et qu'il essaye de se joindre à un autre canal.

406 ERR_WASNOSUCHNICK "<pseudonyme> :Il n'y a pas ce pseudonyme"
- Retourné par WHOWAS pour indiquer qu'il n'y a pas d'informations d'historique pour ce pseudonyme.

407 ERR_TOOMANYTARGETS "<cible> :<code d'erreur> receveurs. <message d'interruption>"
- Retourné à un client qui tente d'envoyer un PRIVMSG/NOTICE en utilisant le format de destination utilisateur@hôte et pour un utilisateur@hôte qui a plusieurs occurrences.
- Retourné à un client qui essaye d'envoyer un PRIVMSG/NOTICE à un trop grand nombre de receveurs.
- Retourné à un client qui tente de se JOINDRE à un canal sûr en utilisant le petit nom alors qu'il y a plus d'un de ces canaux.

408 ERR_NOSUCHSERVICE "<nom de service> :Un tel service n'existe pas"
- Retournée à un client qui tente d'envoyer un SQUERY à un service qui n'existe pas.

409 ERR_NOORIGIN ":Pas d'origine spécifiée"
- Message PING or PONG auquel manque le paramètre d'origine.

411 ERR_NORECIPIENT ":Pas de receveur donné (<commande>)"
412 ERR_NOTEXTTOSEND ":Pas de texte à envoyer"
413 ERR_NOTOPLEVEL "<gabarit> :Pas de domaine de niveau supérieur spécifié"
414 ERR_WILDTOPLEVEL "<gabarit> :Caractère générique dans le domaine de niveau supérieur"

415 ERR_BADMASK "<gabarit> :Mauvais gabarit de serveur/hôte"
- Les erreurs 412 - 415 sont retournées par PRIVMSG pour indiquer que le message n'a pas été livré pour une certaine raison. ERR_NOTOPLEVEL et ERR_WILDTOPLEVEL sont des erreurs qui sont retournées lorsque est tenté un usage invalide de "PRIVMSG $<serveur>" ou "PRIVMSG #<hôte>".

416 ERR_TOOMANYMATCHES "<canal> :Résultat trop long (essayer en local)"
- Retourné par un serveur en réponse à un message LIST ou NAMES pour indiquer que le résultat contient trop d'éléments pour être retourné au client.

421 ERR_UNKNOWNCOMMAND "<commande> :Commande inconnue"
- Retourné à un client enregistré pour indiquer que la commande envoyée est inconnue du serveur.

422 ERR_NOMOTD ":Il manque le fichier MOTD"
- Le fichier MOTD du serveur n'a pas pu être ouvert par le serveur.

423 ERR_NOADMININFO "<serveur> :Pas d'informations administratives disponibles"
- Retourné par un serveur en réponse à un message ADMIN lorsque il ne réussit pas à trouver les informations appropriées.

424 ERR_FILEERROR ":Erreur de fichier en faisant <file op> sur <file>"
- Message d'erreur générique utilisé pour rapporter l'échec d'une opération de fichier durant le traitement d'un message.

431 ERR_NONICKNAMEGIVEN ":Aucun pseudonyme n'est donné"
- Retourné quand un paramètre pseudonyme attendu pour une commande n'est pas trouvé.

432 ERR_ERRONEUSNICKNAME "<pseudonyme> :Pseudonyme erroné"
- Retourné à la réception d'un message NICK qui contient des caractères qui ne rentrent pas dans le jeu défini. Voir au paragraphe 2.3.1 les détails sur les pseudonymes valides.

433 ERR_NICKNAMEINUSE "<pseudonyme> :Pseudonyme déjà utilisé"
- Retourné lorsque le traitement d'un message NICK résulte en une tentative de changer le pseudonyme en un pseudonyme existant actuellement.

436 ERR_NICKCOLLISION "<pseudonyme> :Collision de pseudonymes KILL de <utilisateur>@<hôte>"
- Retourné par un serveur à un client quand il détecte une collision de pseudonymes (enregistrement d'un NICK qui existe déjà par un autre serveur).

437 ERR_UNAVAILRESOURCE "<pseudonyme/canal> :Le pseudonyme/canal est temporairement indisponible"
- Retourné par un serveur à un utilisateur qui essaye de se joindre à un canal actuellement bloqué par le mécanisme de délai de canal.
- Retourné par un serveur à un utilisateur qui essaye de changer de pseudonyme lorsque le pseudonyme désiré est bloqué par le mécanisme de délai de pseudonyme.

441 ERR_USERNOTINCHANNEL "<pseudonyme> <canal> :Ils ne sont pas sur ce canal"
- Retourné par le serveur pour indiquer que l'utilisateur cible de la commande n'est pas sur le canal désigné.

442 ERR_NOTONCHANNEL "<canal> :Vous n'êtes pas sur ce canal"
- Retourné par le serveur chaque fois qu'un client essaye d'effectuer une commande affectant un canal dont il n'est pas membre.

443 ERR_USERONCHANNEL "<utilisateur> <canal> :est déjà sur le canal"
- Retourné quand un client essaye d'inviter un utilisateur sur un canal sur lequel il est déjà.

444 ERR_NOLOGIN "<utilisateur> :Utilisateur non enregistré"
- Retourné par le convoquant après une commande SUMMON pour laquelle un utilisateur a été incapable de se conformer car ils ne sont pas enregistrés.

445 ERR_SUMMONDISABLED ":SUMMON a été désactivé"
- Retourné en réponse à la commande SUMMON. DOIT être retournée par tout serveur qui ne la met pas en œuvre.

446 ERR_USERSDISABLED ":USERS a été désactivé"
- Retourné en réponse à la commande USERS. DOIT être retournée par tout serveur qui ne la met pas en œuvre.

451 ERR_NOTREGISTERED ":Vous n'avez pas été enregistré"
- Retourné par le serveur pour indiquer que le client DOIT être enregistré avant que le serveur lui permette d'être analysé en détail.

461 ERR_NEEDMOREPARAMS "<commande> :Pas assez de paramètres"
- Retourné par le serveur sur de nombreuses commandes pour indiquer au client qu'il n'a pas fourni assez de paramètres.

462 ERR_ALREADYREGISTRED ":Commande non autorisée (déjà enregistrée)"
- Retourné par le serveur à toute liaison qui essaye de changer une partie des détails enregistrés (comme un mot de passe ou des détails d'utilisateur à partir du second message USER).

463 ERR_NOPERMFORHOST ":Votre hôte n'est pas parmi les privilégiés"
- Retourné à un client qui tente de s'enregistrer auprès d'un serveur qui n'a pas été réglé pour permettre des connexions à partir de l'hôte duquel la connexion est tentée.

464 ERR_PASSWDMISMATCH ":Mot de passe incorrect"
- Retourné pour indiquer l'échec d'une tentative d'enregistrement d'une connexion pour laquelle un mot de passe est exigé et qui n'a pas été donné ou est incorrect.

465 ERR_YOUREBANNEDCREEP ":Vous êtes interdit sur ce serveur"
- Retourné après une tentative de connexion et d'enregistrement auprès d'un serveur qui a été réglé pour refuser explicitement les connexions avec le demandeur.

466 ERR_YOUWILLBEBANNED
- Envoyé par un serveur à un utilisateur pour l'informer que l'accès au serveur va bientôt lui être refusé.

467 ERR_KEYSET "<canal> :Clé de canal déjà établie"
471 ERR_CHANNELISFULL "<canal> :Le canal ne peut être joint (+l)"
472 ERR_UNKNOWNMODE "<char> :est un caractère de mode inconnu pour <canal>"
473 ERR_INVITEONLYCHAN "<canal> :Le canal ne peut être joint (+i)"
474 ERR_BANNEDFROMCHAN "<canal> :Le canal ne peut être joint (+b)"
475 ERR_BADCHANNELKEY "<canal> :Le canal ne peut être joint (+k)"
476 ERR_BADCHANMASK "<canal> :Mauvais gabarit de canal"
477 ERR_NOCHANMODES "<canal> :Le canal n'accepte pas ces modes"
478 ERR_BANLISTFULL "<canal> <char> :La liste de canaux est pleine"
481 ERR_NOPRIVILEGES ":Permission refusée – Vous n'êtes pas un opérateur IRC"
- Toute commande exigeant les privilèges d'opérateur pour fonctionner DOIT retourner ces erreurs pour indiquer l'échec de la tentative.

482 ERR_CHANOPRIVSNEEDED "<canal> :Vous n'êtes pas un opérateur du canal"
- Toute commande qui exige les privilèges 'chanop' (comme les messages MODE) DOIT retourner cette erreur si le client qui fait cette tentative n'est pas opérateur de canal sur le canal spécifié.

483 ERR_CANTKILLSERVER ":Vous ne pouvez pas tuer un serveur!"
- Toute tentative d'utiliser la commande KILL sur un serveur doit être refusée et cette erreur retournée directement au client.

484 ERR_RESTRICTED ":Votre connexion est soumise à des restrictions !"
- Envoyé par le serveur à un utilisateur au moment de la connexion pour indiquer la nature restreinte de la connexion (mode d'utilisateur "+r").

485 ERR_UNIQOPPRIVSNEEDED ":Vous n'êtes pas l'opérateur de canal d'origine"
- Tout MODE exigeant les privilèges de "créateur de canal" DOIT retourner cette erreur si le client qui fait la tentative n'est pas opérateur de canal sur le canal spécifié.

491 ERR_NOOPERHOST ":Pas de O-lignes pour votre hôte"
- Si un client envoie un message OPER et si le serveur n'a pas été configuré pour permettre les connexions à partir de l'hôte du client comme un opérateur, cette erreur DOIT être retournée.

501 ERR_UMODEUNKNOWNFLAG ":Fanion MODE inconnu"
- Retourné par le serveur pour indiquer qu'un message MODE a été envoyé avec un paramètre pseudonyme et que le fanion mode envoyé n'a pas été reconnu.

502 ERR_USERSDONTMATCH ":On ne peut pas changer le mode pour d'autres utilisateurs"
- Erreur envoyée à tout utilisateur qui essaye de voir ou changer le mode d'utilisateur pour un utilisateur autre que lui-même.

5.3 Numéros réservés

Ces numéros ne sont pas décrits plus haut car ils ne rentrent dans aucune des catégories suivantes :
1. plus utilisé ;
2. réservé pour une utilisation future prévue ;
3. actuellement utilisé mais faisant partie d'une "caractéristique" générique du serveur IRC actuel.

231	RPL_SERVICEINFO	232	RPL_ENDOFSERVICES	233	RPL_SERVICE
300	RPL_NONE	316	RPL_WHOISCHANOP		
361	RPL_KILLDONE	362	RPL_CLOSING	363	RPL_CLOSEEND
373	RPL_INFOSTART	384	RPL_MYPORTIS		
213	RPL_STATSCLINE	214	RPL_STATSNLINE	215	RPL_STATSILINE
216	RPL_STATSKLINE	217	RPL_STATSQLINE	218	RPL_STATSYLINE
240	RPL_STATSVLINE	241	RPL_STATSLLINE	244	RPL_STATSHLINE
245	RPL_STATSSLINE	246	RPL_STATSPING	247	RPL_STATSBLINE
250	RPL_STATSDLINE				
492	ERR_NOSERVICEHOST				

6. Mises en œuvre actuelles

Le logiciel IRC, version 2.10 est la seule mise en œuvre complète du protocole IRC (client et serveur). À cause de la faible quantité de changements au protocole client depuis la publication de la [RFC1459], les mises en œuvre qui la suivent seront vraisemblablement conformes au présent protocole ou n'auront besoin que de peu de changements pour atteindre la conformité.

7. Problèmes rencontrés

Un certain nombre de problèmes du protocole de client IRC ont été reconnus, et plus généralement avec le protocole de serveur IRC. Afin de préserver la rétro compatibilité avec les anciens clients, le présent protocole n'a presque pas évolué depuis la publication de la [RFC1459].

7.1 Pseudonymes

L'idée du pseudonyme sur IRC est très pratique pour les utilisateurs lorsque ils parlent ensemble en-dehors d'un canal, mais il n'y a qu'un espace fini de pseudonymes et étant donné ce qu'ils sont, il n'est pas rare que plusieurs personnes veuillent utiliser le même pseudonyme. Si un pseudonyme est choisi par deux personnes qui utilisent le présent protocole, l'une d'elles va échouer, ou toutes deux seront retirées par l'utilisation d'une commande KILL par le serveur (voir au paragraphe 3.7.1).

7.2 Limitation des caractères génériques

Il n'y a aucun moyen d'esquiver le caractère "\" (%x5C). Bien que ce ne soit généralement pas un problème, cela rend impossible de former un gabarit avec un caractère barre oblique inverse ("\") précédant un caractère générique.

7.3 Considérations pour la sécurité

Les questions de sécurité qui se rapportent au présent protocole sont exposées dans la [RFC2813] "Protocole de serveur IRC" car elles ne sont un problème que pour le côté serveur de la connexion.

8. Prise en charge et disponibilité actuelle

Listes de diffusion de discussion en rapport avec IRC :
Discussion générale : ircd-utilisateurs@irc.org
Développement de protocole : ircd-dev@irc.org

Mises en œuvre de logiciels :
ftp://ftp.irc.org/irc/server
ftp://ftp.funet.fi/pub/unix/irc
ftp://coombs.anu.edu.au/pub/irc

Groupe de nouvelles : alt.irc

9. Remerciements

Certaines parties du présent document ont été copiées de la [RFC1459] qui a été la première à documenter formellement le protocole IRC. Il a aussi bénéficié de nombreuses séances de révision et de commentaires. En particulier, les personnes suivantes ont fourni des contributions significatives à ce document : Matthew Green, Michael Neumayer, Volker Paulsen, Kurt Roeckx, Vesa Ruokonen, Magnus Tjernstrom, Stefan Zehl.

10. Références

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

[RFC1459] J. Oikarinen et D. Reed, "Protocole Internet de relais de débats", mai 1993. *(Exp., MàJ par 2810-13)*

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

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

[RFC2810] C. Kalt, "Relais pour la causette Internet : architecture", avril 2000. *(Information)*

[RFC2811] C. Kalt, "Relais pour la causette Internet : gestion de canal", avril 2000. *(Information)*

[RFC2813] C. Kalt, "Relais pour la causette Internet : protocole serveur", avril 2000. *(Information)*

11. Adresse de l'auteur

Christophe Kalt
99 Teaneck Rd, Apt #117
Ridgefield Park, NJ 07660
USAmél : kalt@stealth.net

12. Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'éditeur des RFC est actuellement assuré par la Internet Society.