<http://kabehana.hoops.ne.jp/etc/rfc1459j.html>

Request for Comments: 1459

Internet Relay Chat Protocol

[この文章の取り扱い]

この文章にはInternet communityのためのExperimental Protocolを定義し、改善のための意見と提案が寄せられています。このプロトコルと標準化の状況については"IAB Official Protocol Standards"の最新版を参照して下さい。この文章の配布は自由に行ってかまいません。
<訳者注:>これは原文の取り扱いです。日本語訳についてはセクション13.2「日本語訳について」に準じます。また、IRCプロトコルの定義はあくまで原文であることを断っておきます。

[概要]

最初にBBS上のユーザ間のチャットの方法として使われ初めてから4年間以上にわたって、IRCプロトコルは開発が進められています。現在、このプロトコルはサーバ／クライアント方式で世界中のネットワークをカバーしています。そして、ネットワークの成長に伴って、プロトコルも改善されてきています。過去2年間、主要なIRCネットワークを利用するユーザの数は10倍にも膨れ上がっています。IRCプロトコルはテキストベースのプロトコルで、クライアントはサーバとソケットでつなぐシンプルなプログラムです。

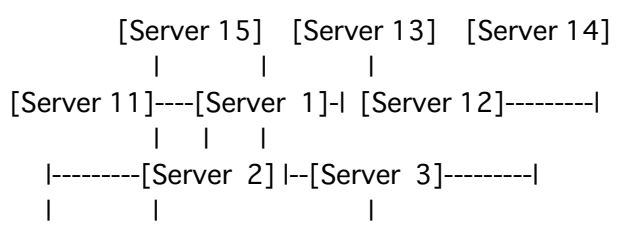
[目次]

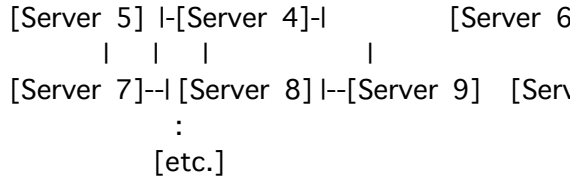
[1. 導入]

IRC(=Internet Relay Chat)プロトコルはテキストベースで会話をするプロトコルとして、何年もに渡ってその設計が改良されてきています。このドキュメントには現在のIRCプロトコルが述べられています。
IRCプロトコルはTCP/IPネットワークプロトコルを使用したシステム上で開発されましたが、これは必要条件ではなく、この動作はTCP/IPプロトコルの範囲に限定するものではありません。
IRCは遠隔的会話システムです。そして、これは(クライアント／サーバモデルのため)多くの機種で動作します。典型的な構成としては、クライアント(や他のサーバ)の接続点の中心として一つのプロセス(サーバ)を配置します。必要なメッセージの配信、多重送信等の機能をサーバが提供します。

[1.1 サーバ]

サーバがIRCのバックボーンを形成しています。サーバは、クライアントが互いに接続して会話をしたり、他のサーバとの接続して会話をする接続点を提供し、IRCネットワークを形成しています。IRCサーバのとりうるネットワーク形態は伸縮自在な木構造のみです。(Fig.1を見よ)あるサーバから見て、それ以外のネットワークの中心ノードとして働くように、各々のサーバを配置します。





[Fig.1 IRCネットワークの形式]

[1.2 クライアント]

一つのサーバに接続しているクライアントは他のサーバに接続してはなりません。
各クライアントは他のクライアントから最長9文字のニックネームによって認識されます。ニックネームにどの文字が使えてどの文字が使えないかはプロトコルの文法規約を参照して下さい。
ニックネームに加えて次に挙げる全クライアントの情報を、すべてのサーバは保持していなくてはなりません:

(a)クライアントの走っているホストの正式名称
(b)そのホスト上でのクライアントのユーザ名
(c)クライアントの接続しているサーバ

[1.2.1 IRCオペレータ]

適切な状態にIRCネットワークを保つため、クライアントの特別な形態であるIRCオペレータ(IRC operator)にネットワーク上で高レベルな保守機能を使用することを許可しています。オペレータに与えられた権限が「危険と」考えることもできますが、それでもなおIRCオペレータは必要とされます。長期間に渡って粗悪なネットワークルーティング(network routing)が使われることを防ぐために、必要に応じて接続を切ったり、つなぎ直したりするネットワークの根幹にかかわる操作をIRCオペレータは行うことができます。プロトコルでは、この必要性を認めているので、IRCオペレータのみにこのような機能を使用する権限を認めています。
セクション4.1.7(SQUITメッセージ)及びセクション4.3.5(CONNECTメッセージ)を参照のこと
まだ多くの議論の余地があるIRCオペレータの権限に、「強制的に」ネットワーク接続からクライアントを排除する能力が挙げられます。オペレータはどんなクライアント-サーバ間の接続でも閉じることができます。乱用すると破壊的になり、悩みのタネになってしまうので、この問題に対する正当な理由はデリケートです。(IRCオペレータはこのような一般クライアントを追放することができます。) このような行動のより詳細については、セクション4.6.1(KILLメッセージ)を参照のこと

[1.3 チャンネル]

チャンネルとは、一つ以上のクライアントから構成された、ある名付けられたグループをいいます。その「チャンネル」に宛られた全てのメッセージはそれを構成するクライアントに送信されます。チャンネルは最初のクライアントがJOIN(参加)したときに暗黙のうちに作り出され、そして最後のクライアントが抜けた時に消滅します。チャンネルが存在する間、クライアントはチャンネルの名前を使って、チャンネルを参照することができます。
チャンネルの名前は最長200文字までの文字列です。(これは'&'または'#'の文字で始まります) 先頭の文字が'&'か'#'であることが必要な条件であることを除くと、スペース(' ')、C-g(^G ASCIIコードで7)とコンマ(',' これはリストのアイテムのセパレータとしてプロトコルが使用します)を含まないということだけが、チャンネル名に対する制約となります。
2種類のチャンネルがこのプロトコルで認められています。一つは'#'ではじまるチャンネルでIRCネットワークに接続するすべてのサーバに対して公開されているチャンネルです。この種のチャンネルの第一の特徴ははチャンネルにJOINしているサーバ上のクライアントが一つでもあれば、IRCネットワーク上に存在するということです。これらは'&'の文字で始まるものとは区別されます。これら2種類のチャンネル

上で利用できる種々のチャンネルモードで個々のチャンネルの性質を変更することができます。
<訳者注:>&ではじまるチャンネルはサーバにローカルなチャンネルです。(つまり同じサーバにつないでるクライアントからしか見えない)
このメッセージの詳細はセクション 4.2.3(MODEメッセージ)を参照のこと
新しいチャンネルを作るもしくは現存のチャンネルに加わるには、クライアントはチャンネルにJOINすることが必要です。もしチャンネルにJOINするよりも前にチャンネルが存在していなければ、チャンネルが作られ、そして作ったクライアントがチャンネルオペレータになります。チャンネルがすでに存在していた場合、そのチャンネルにJOINする要求が認められるかどうかは、その時のチャンネルのチャンネルモードによります。例えば、チャンネルモードが"invite-only"(mode +i)ならば、"INVITE(招待)"されている時だけJOINできます。
クライアントは同時に複数のチャンネルに参加できます。しかし、初心者からベテランまで十分な数として、10チャンネルまでに制限されています。
この点のより詳細はセクション8.13を参照のこと。
2つのサーバ間が分断され、IRCネットワークの接続が切れた場合、双方が、まだ接続されているサーバ内でチャンネルを再構成します。ひょっとすると分断されたどちらか一方の側では消滅するかもしれません。分断が回復したとき、接続したサーバは互いに、それぞれのチャンネルとそのチャンネルモードをアナウンスします。チャンネルが双方で存在していた場合、JOINとMODEは新たに接続した双方のチャンネルにどのクライアントがいてどのようなモードなのか整合するように包括的な方法で解決されます。

[1.3.1 チャンネルオペレータ]

権限を与えられたチャンネル上のチャンネルオペレータ("chop"や"chanop"とも呼ばれています。※日本では「なると持ち」と呼ばれています)がそのチャンネルを「所有」するとみなされます。チャンネルオペレータのこの地位を認めて、チャンネルの管理と健全性を保つための権限をチャンネルオペレータは持っています。チャンネルオペレータは、チャンネルの所有者なので、チャンネルオペレータに行動の理由を問うことはできません。もし仮にチャンネルオペレータの行動が一般社会常識に反したり、口汚くてもです。この時はIRCオペレータに仲裁するように頼むか、あるいはユーザが直ちに去り、問題のチャンネルオペレータの所有しているチャンネルから他の場所へ行き、自分自身でチャンネルを作るのがよいでしょう。
チャンネルオペレータのみが使用できるメッセージは以下のものです:

KICK - クライアントをチャンネルから追放します
MODE - チャンネルモードを変更します
INVITE - "invite-only"(mode +i)のチャンネルにクライアントを招待します
TOPIC - (mode +t)のときにチャンネルを変更します

チャンネルに参加しているときに(NAMES,WHO,WHOISメッセージのリプライとして表示される)ニックネームのそばの'@'記号でチャンネルオペレータは識別できます。(※'@'記号がラーメンなどに浮いている「なると」に似ている事が、日本ではチャンネルオペレータの事を「なると持ち」という由来です。)

[2. IRCの仕様]

[2.1 概観]

ここに記述されているプロトコルはサーバ-サーバ間、クライアント-サーバ間の接続の両方共で使用されています。しかし、サーバの接続に対する制限よりクライアントの接続に対する制限の方が厳しくなっています。クライアントの接続は信頼性がないとみなさているからです。

[2.2 文字コード]

具体的な文字セットの仕様は定められていません。1オクテット(=8 bit)のコードの

セットをプロトコルではベースとしています。このオクテットをいくつかまとめたものから、各メッセージは成り立っています。オクテットの値のうちいくつかはデリミタとして働くコントロールコードに使われています。
8ビットプロトコルであることは問題になりません。デリミタとキーワードは大抵のUSASCIIターミナルやtelnet接続のプロトコルのようなものだからです。
スカンジナビア語がIRCの起源なので、文字'{}|'はそれぞれ文字'[]\'の小文字に当たるものとみなされます。これは2つのニックネームが同じのものであることを確定するときに重要な問題となります。

[2.3 メッセージ]

サーバとクライアントは互いにメッセージを送信し合います。これらのうちには、リプライするものもしないものもあります。メッセージが正しいコマンドを含んでいた場合、後のセクションで述べますが、クライアントは仕様通りのリプライを期待しています。しかしこれはリプライを永久に待たねばならないことを意味しているわけではありません。クライアント-サーバ間、サーバ-サーバ間の通信は基本的に非同期です。
どのIRCメッセージも最大3つの主な部分から成り立っています:

(a)プレフィックス(prefix)(使用は任意)
(b)コマンド(command)
(c)コマンドパラメータ(command parameters)(最大15個まで)

プレフィックス、コマンド、全コマンドパラメータは1つ以上のASCIIコードのスペース文字(0x20)によって分けられます。
先頭に付いてるコロン(':',0x3b)一文字によって、プレフィックスの存在が表現されます。そしてこれはメッセージ自体の先頭の文字となります。コロンとプレフィックスの間には飛び(空白文字)があってはいけません。プレフィックスはメッセージの送信元を表すために、サーバが使用します。プレフィックスがメッセージから抜け落ちていた場合、送信してきた接続先が送信者であると仮定します。クライアントはメッセージを送信するときにプレフィックスを使用すべきではありません。クライアントがプレフィックスを使用するとすれば、唯一の妥当なプレフィックスはそのクライアントと関連付けて登録されているニックネームのみです。プレフィックスによって識別された送信者がサーバの持っているデータベースで見つからない、または、メッセージの来た接続先とは異なった接続のものとしてその送信者が登録されている場合、サーバは黙ってそのメッセージを無視します。
コマンドは正しいIRCコマンドまたはASCIIテキストを表された3桁の10進数のどちらかでなくてはいけません。
IRCメッセージはいつもCR-LF(Carriage Return- Line Feed)で終わる文字の列です。そして、これらのメッセージは512文字を超えた長さにはできません。全文字数に末尾のCR-LFは含まれます。従って、最大510文字がコマンドとそのパラメータに使用できます。連続したメッセージ行のための規定はありません。
この事のより詳細についてはセクション7を参照のこと。

[2.3.1 疑似BNFによるメッセージ形式]

プロトコルのメッセージを連続したオクテットのストリームから抽出しなくてはいけません。現在の解決策はメッセージの区切りとして、2文字(CRとLF)を使用することです。空のメッセージは黙って無視されます。特別な問題が無ければメッセージの間にCR-LFシーケンスを使用できます。
抽出されたメッセージは<prefix>,<command>,<list of parameters>の成分に構文解析され、さらに、<list of parameters>は<middle>または<tailing>成分にマッチします。このためのBNF表現は以下にあげるようなものになります。

```
<message>  ::= [':' <prefix> <SPACE> ] <command> <params> <crlf>
<prefix>   ::= <servername> | <nick> [ '!' <user> ] [ '@' <host> ]
<command>  ::= <letter> { <letter> } | <number> <number> <number>
```

```
<SPACE>    ::= ' ' { ' ' }
<params>   ::= <SPACE> [ ':' <trailing> | <middle> <params> ]

<middle>   ::= <先頭が':'ではなく,SPACE,NUL,CR,CFを含まない、空でないオクテットの列>
<trailing> ::= <SPACE,NUL,CR,CFを含まないオクテットの列(空のオクッテトの列も可)>
<crlf>     ::= CR LF
```

注:
1)
<SPACE>は1つ以上の空白文字(' ',0x20)だけからなります。タブおよび、その他全てのコントロール文字は「非空白文字」であることに特に注意して下さい。
2)
パラメータリストを抽出した後は<middle>,<trailing>のいずれにマッチしても、全てのパラメータは同等のものとなります。 <trailing>は単にパラメータの範囲内でSPACEを許すためのシンタックス上のトリックです。
3)
メッセージフレームの仕様のため、CRとLFがパラメータの文字列の中に入れることができないという事になっています。これはいずれ変更されるかもしれません
4)
NUL文字はメッセージフレームの中では特別なものではありません。そこで、基本的には、NUL文字はパラメータ内の終端に使用できます。しかし、これは普通のCの文字列操作で余計な複雑さを引き起こすことになります。したがって、メッセージの中で使用することを認めてません。
5)
最後のパラメータは空のオクテットの列でもかまいません。
6)
抽出したプレフィックスに使う(['!' <user>] ['@' <host>])はサーバからサーバへの通信には使用してはいけません。これはサーバ-クライアント間でメッセージをクライアントに送信する時に、サーバに追加して要求する必要なしにメッセージがどこから来たかという情報をクライアントに渡す有用な方法として使用されます。
ほとんどのプロトコルメッセージはリスト中の位置からパラメータ文字列を抽出するにあたり追加の意味と構文を定義しています。 例えば、多くのサーバコマンドはコマンドの後の最初のパラメータが送信先のリストであることを仮定するでしょう。
それは以下のように表現することができます:

```
<target>    ::= <to> [ "," <target> ]
<to>        ::= <channel> | <user> '@' <servername> | <nick> | <mask>
<channel>   ::= ('#' | '&') <chstring>
<servername> ::= <host>
<host>      ::=※ホスト名についての詳細はRFC 952 [DNS:4] を参照
<nick>      ::= <letter> { <letter> | <number> | <special> }
<mask>      ::= ('#' | '$') <chstring>
<chstring>  ::= <8bitコード(SPACE, BELL, NUL, CR, LF, ','を除く)>
```

他のパラメータの構文は

```
<user>      ::= <nonwhite> { <nonwhite> }
<letter>    ::= 'a' ... 'z' | 'A' ... 'Z'
<number>    ::= '0' ... '9'
<special>   ::= '-' | '[' | ']' | '\' | '`' | '^' | '{' | '}'
<nonwhite>  ::= <8bitコード(SPACE(0x20),NUL(0x0),CR(0xd),LF(0xa)を除く)
```

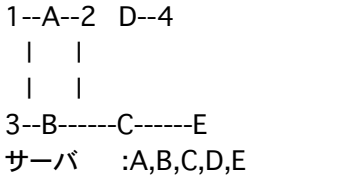
[2.4 ニュメリックリプライ]

サーバに送信された大抵のメッセージが、なんらかのリプライを生成します。もっとも一般的なリプライはニュメリックリプライ(numeric reply)です。これはエラーにも通常のリプライにも使用されます。 ニュメリックリプライは送信者のプレフィックスと3桁の数字、そしてリプライの送信先から成り立っています。 クライアント

はニュメリックリプライを送信することは認められていません。どんなメッセージでもサーバは受け取りますが黙って落とされます。これ以外の点おいては、ニュメリックリプライが普通の文字列より短い3桁の数字からキーワードから成り立っていることを除いて、普通のメッセージとまったく同じように扱われます。種々のリプライのリストがセクション6にあります。

[3. IRCのコンセプト]

このセクションではIRCプロトコルの構成の背景にあるコンセプトと、現在、いろいろな異なる形態のメッセージの送信がどのように実装されているかを述べていきます。



[Fig.2. 小規模なIRCネットワークの例]

[3.1 1=>1の通信]

ほとんどのサーバ-サーバ間のトラフィックは互いのサーバの通信結果だけでは済まないので、1対1の通信は通常、クライアントだけが行います。クライアントが互いに会話をする機能を安全な方法で提供するには、全サーバがどのクライアントにもたどり着けるように、正確に木構造のネットワークに沿って、単一の方向にメッセージを配信できるようになっていなくてはなりません。送信されるメッセージの経路は木構造のネットワーク上の2点間の最も短い経路になります。
以下の例はFig.2に沿って述べられています。

例 1:
クライアント1と2の間のメッセージはサーバAだけを通過します。サーバAはこれを直接クライアント2に送信します。
例 2:
クライアント1と3の間のメッセージはサーバA,Bおよびクライアント3を通過します。他のクライアントとサーバはメッセージを見ることはできません。
例 3:
クライアント2と4の間のメッセージはサーバA,B,C,Dおよびクライアント4を通過します。

[3.2 1=>多人数]

IRCの主な目的は簡単で効果的なコンファレンス(一対多の会話)が可能なフォーラムを提供することです。IRCはこれを達成するため複数の方法を提供しています。それぞれの使用目的によってそれぞれの送信方法があります。

[3.2.1 =>リスト]

一対多の会話の最も効率の悪いスタイルはクライアントがクライアントのリストを使用して通信することです。それがどのようになされるかは、ほとんど自明でしょう。クライアントはメッセージの送信先のリストを与えてメッセージを送信します。サーバはリストを分解し、それぞれの送信先に送信するため、メッセージをコピーして別々に送信します。これはグループを使うより効率的ではありません。なぜなら送信先のリストが分解され、各経路で複製が送信されないという信頼性がチェックをすることなく、送信されるからです。

[3.2.2 =>グループ(チャンネル)]

IRCのチャンネルはマルチキャストグループと等価な役割を持ちます。この存在はダイナミックなものです。(人がチャンネルに参加したり、別れていくにつれて、出現したり消滅したりします。)そして、チャンネルでなされた実際の会話はそのチャンネルのクライアントを受け持っているサーバのみに送信されます。同じチャンネルのクライアントが同一サーバ上に多数いる場合、メッセージテキストはそのサーバにたった一度だけ送信されます。そしてサーバは同じチャンネル上の各クライアントに送信します。各クライアント-サーバ間の送信は繰り返され、送信されたメッセージは各チャンネルのメンバーに到達します。Fig 2.に沿って例を挙げていきます。

例 4:
1つしかクライアントのいないチャンネルでは、チャンネルへのメッセージはサーバへ送信され、そしてその後はどこにも送信されません。
例 5:
チャンネルに2つのクライアントがいるとき、全てのメッセージは、あたかもそれが2つのクアイアント間のプライベートメッセージであるかのように経路を通過します。
例 6:
クライアント1,2,3がチャンネルにいるとき、チャンネルへの全てのメッセージは、それがあたかも一つのクライアントへのプライベートメッセージであるかのように、(チャンネルにいる)各クライアントとメッセージが通過しなくてはならないサーバに送信されます。クライアント1がメッセージを送信したとしましょう。クライアント2にはサーバAから送信され、クライアント3にはサーバBを経由して送信されます。

[3.2.3 ホスト／サーバ マスクに対する通信]

関係ユーザ、ホスト、サーバへのマスクメッセージを使って配信するためのメカニズムをIRCオペレータは利用することができます。これらのメッセージはホストやサーバの情報がマスクに一致するクライアントに送信されます。メッセージはチャンネルと似通った方法でクライアントの所に送信されます。

[3.3 1=>全体]

「1=>全体」タイプのメッセージはブロードキャストメッセージと言った方がよいでしょう。全てのクライアントやサーバまたはその両方にメッセージを送信します。クライアントとサーバからなる巨大なネットワーク上では、一つのメッセージが、全ての送信先に到達する作業のために、ネットワークに大量のトラフィックを生むことになります。

いくつかのメッセージは全サーバへブロードキャストする以外の選択ができません。なぜなら、サーバ間で整合性ある状態情報を、全サーバが保持せねばならないからです。

[3.3.1 クライアント-クライアント]

ある単独のクライアントが他の全クライアントにメッセージが送信するような、メッセージ形態は用意されていません。

[3.3.2 クライアント-サーバ]

大部分の状態情報(チャンネルメンバー,チャンネルモード,ユーザ情報など、)を変化させるコマンドはデフォルトで全サーバに向けて送信せねばなりません。クライアントはこの送信の設定を変更することはできません。

[3.3.3 サーバ-サーバ]

サーバ間のメッセージが全ての「他の」サーバに送信される時は大抵、クライアント、チャンネルあるいはサーバに影響するメッセージの時だけに必要になります。

これはIRCにおいて根幹にかかわる事ですので、サーバから送信されるおおよそ全て
のメッセージは接続している他の全サーバへのブロードキャストとなります。

[4. メッセージの詳細]

以下ではIRCサーバとクライアントが使用可能な各メッセージについて述べていま
す。本プロトコルによる全サーバがこのセクションで述べている全てのコマンドを
実装しなくてはなりません。
一覧表にあるリプライ"ERR_NOSHUCHSERVER"は<server>パラメータに該当するサーバ
が見つからないことを意味しています。このリプライを受けた後は、サーバはその
コマンドのために他のリプライを送信してはいけません。
クライアントと接続しているサーバは完全なメッセージをパースし適切なエラーを返
すことが必要です。サーバがメッセージをパースしているときに致命的なエラーを
発見した場合、エラーをクライアントに返信し、パースを終了しなくてはなりませ
ん。不正なコマンド、サーバに不明なホスト(サーバ,ニックネーム,チャンネル名,
などこのカテゴリに属するもの)、パラメータの不足、特権違反等が致命的エラーと
みなされます。
パラメータが完全にそろっていた場合は、クライアントに返信されてきた応答の正当
性と妥当性をチェックしなくてはなりません。コンマを項目のセパレータとするメ
ッセージの場合、リプライは各項目にたいして送信する必要があります。
この下の例では、いくつかのメッセージはフルフォーマットで使用してます:

    :Name COMMAND parameter list

この例はサーバ間で通過する"Name"からのメッセージを表しています。ここで重要な
点は、正確な経路を通ってリモートサーバへリプライを返すために、メッセージの最
初の送信者の名前が含まれているという点です。

[4.1 接続の開始]

ここで説明されるメッセージはIRCサーバに接続を開始したり、もちろんクライアン
トやサーバが正常に接続を切ったりすることにも使用します。
PASSコマンドはクライアントやサーバの接続の登録に必須ではありません。しかし行
うならば、SERVERメッセージや、すぐ後で説明するNICK/USERメッセージに先だって
行わなければなりません。サーバ間の接続に関しては実際の接続で一定のセキュリ
ティの水準を保つために、パスワードを設定することを強く推奨します。クライア
ントが登録を行う場合、以下に述べる順にメッセージを送ることを推奨します:

    1. PASSメッセージ
    2. NICKメッセージ
    3. USERメッセージ

[4.1.1 PASS(password)メッセージ]

    Command: PASS
    Parameters: <password>

PASSメッセージは接続パスワードを送信するために使います。パスワードは接続を
開始しようとする前に送信することが可能ですし、送信しなくてはなりません。現
在クライアントは、NICK/USERの関連付けを送信する前にPASSメッセージを送信する
こと、サーバはSERVERメッセージを送信する前に必ずPASSメッセージを送信せねばな
らないとが、定められています。送信されたパスワードはC/N行(サーバ)またはI行
(クライアント)にマッチしなくてはなりません。接続が登録される前に何度PASSメ
ッセージを送信してもかまいませんが、最後に送信したものだけが認証には使用され
ます。接続が一度登録されたら変更することはできません。

ニュメリックリプライ:

ERR_NEEDMOREPARAMS
ERR_ALREADYREGISTRED

例:
PASS ここに秘密のpasswordを書く

[4.1.2 NICK(nickname)メッセージ]

Command: NICK
Parameters: <nickname> [<hopcount>]

NICKメッセージはクライアントにニックネームを与えたり、前のものから変更したりするために使用します。 <hopcount>のパラメータはニックネームのホームサーバからどのくらい離れているか示すためにサーバだけが使用します。ローカルな接続のhopcountは0です。クライアントが送った場合は、hopcountは無視すべきです。
サーバが受信したNICKメッセージが他のクライアントのすでに使用しているニックネームと同一であることが判明した場合、ニックネームの衝突が発生します。ニックネームの衝突のがおこると、ニックネームに関する全ての情報がサーバのデータベースから取り除かれます。そして他の全サーバのデータベースからこのニックネームを取り除くために、KILLメッセージが送信されます。衝突を引き起こしたNICKメッセージがニックネームの変更ならば、そのときは元の(古い)ニックネームもまた削除されます。
サーバがデータベースにあるのと同一のニックネームを指定するNICKメッセージを直接接続してきたローカルなクライアントから受け取った場合は、そのローカルなクライアントに対してERR_NICKCOLLISIONを発行し、NICKメッセージをドロップします。そしてニックネームは生成されず、KILLも行われません。

ニュメリックリプライ:
ERR_NONICKNAMEGIVEN
ERR_ERRONEUSNICKNAME
ERR_NICKNAMEINUSE
ERR_NICKCOLLISION

例:
NICK Wiz
新しいニックネーム"Wiz"を登録します。
:WiZ NICK Kilroy
WiZはニックネームをKilroyに変更します

[4.1.3 USERメッセージ]

Command: USER
Parameters: <username> <hostname> <servername> <realname>

USERメッセージは接続の最初に新しいクライアントのユーザ名、ホスト名、サーバ名、本名を指定するために使います。これは2つのサーバ間での通信でもIRC上の新しいクライアントの到着を示すのに使用します。USERとNICKの両方を受け取った後にサーバはユーザを登録するからです。
サーバ間ではUSERメッセージはクライアントのニックネームのプレフィックスが必要です。 USERメッセージは直接接続したクライアントから来た時、<hostname>と<servername>をIRCサーバは(安全のために)無視することに注意してください。しかし、これらはサーバ-サーバ通信では使用されます。 新しいユーザを残りのネットワークに紹介する時、USERコマンドで情報を送信する前にNICKメッセージを送信せねばならないことを意味します。
<realneme>のパラメータは最後のパラメータであることに注意してください。これは

このパラメータはスペース文字を含むかもしれないからです。このようにスペースを
認識させるためにコロン(':')のプレフィックスを付けねばなりません。
<username>の取得はUSERメッセージにのみ依存しているので、クライアントは偽るこ
とが容易です。そこでサーバには"Indentity Server"の使用が推奨されています。
(訳者注:RFC1413を参照) クライアントの接続しているホストが"Identity Server"を
使用可能ならば<username>は"Identity Server"からのリプライに従って設定されま
す。

ニュメリックリプライ:
ERR_NEEDMOREPARAMS
ERR_ALREADYREGISTRED

例:

USER guest tolmoon tolsun :Ronnie Reagan

"guest"という<username>で<realname>が"Ronnie Reagan"をクライアントに登録しま
す。

:testnick USER guest tolmoon tolsun :Ronnie Reagan

サーバ間で送信されるUSERメッセージの送信者のニックネームつきのメッセージで
す。

[4.1.4 SERVERメッセージ]

Command: SERVER
Parameters: <servername> <hopcount> <info>

SERVERメッセージは新しく接続した端点はサーバであることを伝えるのに使用しま
す。このメッセージは全ネットワークにサーバのデータを伝えるためにも使われま
す。新しいサーバがネットワークに接続したとき、このサーバについての情報はネ
ットワーク全体にブロードキャストされます。<hopcount>はそのサーバからどのく
らい離れているかという内部情報を全てのサーバに渡すのに使われます。完全なサ
ーバリストを漬かって、全体のサーバツリーのマップを構築することが可能となって
います。しかし、ホストマスクがこれを行うのに邪魔になります。
SERVERメッセージは以下のどちらかの場合の時しか受け付けてはなりません。:

(a) 接続が確立していてかつサーバとして接続しようとしている場合
(b) 既存の接続を通して新しいサーバを接続しようとしている場合

(b)の場合、新しいサーバはSERVERメッセージを使って既存の接続先の SERVERメッセ
ージの受信時に起こるほとんどのエラーは受信先のホスト(ターゲットSERVER)が接続
を拒否した結果です。エラーリプライは通常、ニュメリックリプライよりむしろ
"ERROR"メッセージを使って送信されます。これは、この場合には有益な性質を持っ
ているためです。
SERVERメッセージが解析された結果、すでに接続されているサーバへの登録が試みら
れた場合、サーバへの2重の経路が形成され、非循環的なIRCツリーが壊されるので、
そのメッセージによる接続は(正しい手続きの後に)閉じなくてはなりません。

ニュメリックリプライ:
ERR_ALREADYREGISTRED

例:

SERVER test.oulu.fi 1 :[tolsun.oulu.fi] Experimental server

新しいサーバtest.oulu.fiを自分自身を登録し、接続の確立を試みます。[]の中の
名前はtest.oulu.fiの走っているホストのホスト名です

:tolsun.oulu.fi SERVER csd.bu.edu 5 :BU Central Serve

サーバ tolsun.oulu.fiは5 hop 離れたcsd.bu.eduに接続します。

[4.1.5 OPER(operator)メッセージ]

Command: OPER
Parameters: <user> <password>

OPERメッセージは一般クライアントがIRCオペレータの権限を得るために使用します。<user>と<password>の組み合わせがIRCオペレータの権限を得るために必要です。
OPERメッセージを送信したクライアントがそのユーザに割り当てられた正しいパスワードを送った場合、サーバはクライアントのニックネームに対して"MODE +o"を送信することによって新しいオペレータを残りのネットワークに知らせます
OPERメッセージはクライアント-サーバ間のみで使用されます。

ニュメリックリプライ:
ERR_NEEDMOREPARAMS
ERR_NOOPERHOST
ERR_PASSWDMISMATCH
RPL_YOUREOPER

例:

OPER foo bar

"foo"というユーザ名で"bar"をpasswordとしてIRCオペレータに登録することを試みます

[4.1.6 QUITメッセージ]

Command: QUIT
Parameters: [<quit message>]

クライアントのセッションを<quit message>と共に終了します。サーバはQUITメッセージを送信したクライアントとの接続を閉じなくてはなりません。<quit message>が与えられた場合、これはデフォルトメッセージのニックネームの代わりに送信されます。
ネットスプリット(2つのサーバ間の接続の切断)が発生したとき、<quit message>はネットスプリットに巻き込まれた2つのサーバの名前をスペースで区切ったものとなります。1番目の名前はまだ接続しているサーバのもので、2番目の名前は接続が切断されたサーバの名前です。他の何らかの理由で、クライアントがQUITメッセージを発行することなしにクライアントの接続が閉じたられた場合(例えばクライアントが死んで、ソケット上にEOFが発生した時)、サーバは、終了の原因を反映するような終了メッセージを代わりに使用することが必要となります。

ニュメリックリプライ:
存在しません

例:
QUIT :昼めし食いに行きます!
このようなメッセージフォーマットを使います。

[4.1.7 SQUIT(server quit)メッセージ]

Command: SQUIT
Parameters: <server> <comment>

SQUITメッセージはサーバが終了または死ぬことを送信するために使用されます。サーバがあるサーバとの接続を破棄したい場合、サーバは他のサーバにSQUITメッセージを送らねばなりません。その時、<server>パラメータには送り先のサーバの名前を使用します。すると送り先のサーバは終了しようとしているサーバとの接続を閉じます。このメッセージはIRCオペレータがIRCサーバの適切なネットワークを維持するために使用されます。IRCオペレータはリモートサーバの接続に対するSQUITメッセージを発行することも可能です。この場合以下に述べるように、SQUITはIRCオペレータとリモートサーバの間にはさまれる各サーバによって構文解析され、各サーバが保持するネットワークの形状が更新されます。
IRCオペレータは、リモートサーバ(これは現在オペレータのいるサーバ サーバもまた<comment>にエラーやそれに類似したメッセージを置いてゆきます。ネットワークの閉じる接続の向こう側に広がる全サーバのために、閉じる接続の両端のサーバはSQUITメッセージを(そのサーバに接続している他の全てのサーバに)送信することが要求されます。
同様に、切りはなされる方のネットワークに接続している全クライアント分のQUITメッセージを送信せねばなりません。これに加えて、分断によってそのチャンネルのメンバーを失う全チャンネルのメンバーはQUITメッセージを送信せねばなりません。
サーバとの接続が突然切れた場合(例えば、接続先のサーバが死んだ場合)、この接続の切断を検知したサーバは接続が閉じられた残りのネットワークに、何らかの妥当なコメントをコメントフィールドに記載して知らせることが必要です。

ニュメリックリプライ:
ERR_NOPRIVILEGES
ERR_NOSUCHSERVER

例:

SQUIT tolsun.oulu.fi :Bad Link ?

サーバリンクtolson.oulu.fi は"Bad link"のために接続を終了しました

:Trillian SQUIT cm22.eng.umd.edu :Server out of control

"Server out of control"が原因でネットワークと"cm22.eng.umd.edu"の接続が切断されることついてのTrillianからのメッセージです

[4.2 チャンネル操作]

このメッセージ群はチャンネルの操作、チャンネルの性質(チャンネルモード),そしてチャンネルの(特にクライアントの)内容と関係があります。ネットワークで2つのクライアントが最終的に衝突するメッセージを送信した時、これに対する反応には必然的にいく種類かの状態が存在します。いつでも<nickname>が指定できることを保証するためにサーバがニックネームのヒストリーを保持することもまた必要です。
ニックネームが変更された場合、サーバはこのヒストリをチェックします。

[4.2.1 JOIN(参加)メッセージ]

Command: JOIN
Parameters: <channel>{,<channel>} [<key>{,<key>}]

JOINメッセージはクライアントが実際にチャンネルに接続しはじめるのに使います。
クライアントがそのチャンネルにJOINできるかどうかは、クライアントが接続してい
るサーバがチェックします。そのサーバがメッセージを受領した場合、他のサーバは
自動的にクライアントをチャンネルに加えます。
以下のような事が影響します:

1. クライアントはチャンネルがinvite-onlyの時はINVITE(招待)されていなく
てはなりません。
2. クライアントのニックネーム/ユーザ名/ホスト名がアクティブなbanにマッ
チしてはいけません。
3. key(password)が設定されているならば正しくそれを入力しなくてはいけま
せん。

より詳細な記述はMODEメッセージの解説のところにあります。(詳細はセクション
4.2.3を参照のこと。)
まず、クライアントがチャンネルにJOINすると、クライアントにはサーバが受け取っ
たチャンネルに影響する全てのメッセージについてのNOTICEメッセージが送信されて
きます。PRIVMSG/NOTICEメッセージはもちろん,MODE,KICK,PART,QUITメッセージな
どが含まれます。JOINメッセージは各サーバがチャンネル上のクライアントがどこ
にいるか知らなくてはいけないので、全てのサーバへのブロードキャストが必要で
す。これによって、チャンネルへのPRIVMSG/NOTICEメッセージの送信が最適化され
ます。
JOINが成功した場合、クライアントにはチャンネルのトピック(RPL_TOPICが使用され
ます)、そしてチャンネル上のクライアントのリスト(RPL_NAMREPLYが使用されます)
が送られてきます。リストにはJOINしたクライアントも含まれます。

ニュメリックリプライ:
ERR_NEEDMOREPARAMS
ERR_BANNEDFROMCHAN
ERR_INVITEONLYCHAN
ERR_BADCHANNELKEY
ERR_CHANNELISFULL
ERR_BADCHANMASK
ERR_NOSUCHCHANNEL
ERR_TOOMANYCHANNELS
RPL_TOPIC
RPL_NAMEREPLY

例:

JOIN #foobar

チャンネル#fooberにJOINします

JOIN &foo fubar

チャンネル&fooにkey "fubar"を使ってJOINします。

JOIN #foo,&bar fubar

チャンネル#fooにkey"fuber"を使い、&barにはkeyを使わないでJOINします。

JOIN #foo,#bar fubar,foobar

チャンネル#fooにはkey "fubar"をチャンネル#barにはkey"foobar"を使ってJOINします。

JOIN #foo,#bar

チャンネル#fooと#barにJOINします。

:WiZ JOIN #Twilight_zone

WiZからのJOINメッセージ

[4.2.2 PART(離脱)メッセージ]

Command: PART
Parameters: <channel>{,<channel>}

PARTメッセージは<channel>パラメータにリストで指定した全チャンネルからクライアントが抜けるときに使用されます。<channel>パラメータにリストされた全チャンネルのユーザリストからそのクライアントを取り除きます。

ニュメリックリプライ:
ERR_NEEDMOREPARAMS
ERR_NOSUCHCHANNEL
ERR_NOTONCHANNEL

例:

PART #twilight_zone

チャンネル#twilight_zoneから抜けます

PART #oz-ops,&group5

チャンネル&group5と#oz-opsの両方から抜けます

[4.2.3 MODEメッセージ]

Command: MODE

MODEメッセージはIRCでは2つの目的を持つメッセージです。これはユーザ名とチャンネルの両方をモード変更の対象としています。この理論的根拠は、将来ニックネームが使用されなくなったときにそれに等価なものがチャンネルになるからです。MODEメッセージを構文解析する時、最初にメッセージ全体を解析することが推奨されます。そしてモードを変更する際はその解析結果をもとにします。

[4.2.3.1 チャンネルモード]

Parameters: <channel> {[+|-]|o|p|s|i|t|n|b|v} [<limit>] [<user>][<ban mask>]

MODEメッセージはチャンネルオペレータが「彼らの」チャンネルの特徴を変更するために提供されています。チャンネルオペレータがモードを変更できるように、サーバも変更することができる必要があります。

チャンネルで利用できる各種のモードは以下の通りです:

o - チャンネルオペレータの特権(channel Operator privileges)を授与／剥奪します。
p - プライベートチャンネル(Private channel)属性を設定／解除します。
s - シークレットチャンネル(Secret channel)属性を設定／解除します。
i - インバイトオンリー(Invite-only)属性を設定／解除します。
t - トピックの変更権(Topic settable)をチャンネルオペレータのみに設定／解除します。
n - チャンネル外のクライアントのメッセージ(No message)の受信を許可／不許可します。
m - モデレートチャンネル(Moderated channel:司会つきチャンネル)属性を設定／解除します。
l - チャンネルに参加するクライアントの数を制限(user Limit)する。
b - クライアントの侵入を拒むBANマスク(Ban mask)を設定／解除します。
v - モデレートチャンネル属性のついているチャンネルでの発言権を授与／剥奪します。
k - チャンネルキー(channel Key)(=password)を設定／解除します。

'o'と'b'のオプションを使用した時、全部で3つの事がMODEメッセージでは必要とされます。これは、'o'や...と関連づけられた...
<訳者注:>原文も途中で終わっています。ここは「これは、'o'に関連付けられるニックネームや'b'に関連付けられるバンマスクです。」ではないかと思われます。

[4.2.3.2 ユーザモード]

Parameters: <nickname> {[+|-]|i|w|s|o}

ユーザモードは大抵、クライアントが他の人からの見え方やクライアントに送られてくる「余分な」メッセージに影響します。メッセージの送信者とパラメータとして与えられたニックネームの両方が同一の時に、ユーザMODEメッセージは受領されます。
以下に述べるものがモードとして使用可能です:

i - クライアントを不可視(Invisible)とします。
s - サーバのNOTICE(Server notice)メッセージを受信します。
w - クライアントはWALLOPSメッセージを受信します
o - IRCオペレータフラグ(Operator flag)を立てます。

後ほど追加のモードが利用できるようになるかもしれません。
クライアントが自分自身を"+o"フラグを使ってIRCオペレータに任命しようとした場合、その試みは無視されます。しかしながら、自分自身を"-o"を使うことでIRCオペレータをやめること(deopping)することにはなんの制約もありません。

ニュメリックリプライ:
ERR_NEEDMOREPARAMS
ERR_CHANOPRIVSNEEDED
ERR_NOSUCHNICK
ERR_NOTONCHANNEL
ERR_KEYSET
ERR_UNKNOWNMODE
ERR_NOSUCHCHANNEL
ERR_USERSDONTMATCH
ERR_UMODEUNKNOWNFLAG
RPL_CHANNELMODEIS
RPL_BANLIST

RPL_ENDOFBANLIST
RPL_UMODEIS

例:

チャンネルモードの使い方:

MODE #Finnish +im

#Finnish チャンネルにモデレートとインバイトオンリーの属性を付加します。

MODE #Finnish +o Kilroy

#Finnish チャンネル上でのチャンネルオペレータの権限をKilroyさんに授与します。

MODE #Fins -s

#Fins チャンネルからシークレットフラグを解除します。

MODE #42 +k oulu

チャンネルキー(=password)に"oulu"を設定します。

MODE #eu-opers +l 10

チャンネル上のクライアントの数の制限を10に設定します。

MODE &oulu +b

チャンネルに設定されているbanマスクのリストを表示します。

MODE &oulu +b *!*@*

全てのクライアントの参加を拒否します。

MODE &oulu +b *!*@*.edu

ホスト名が"*.edu"にマッチするクライアントの参加を拒否します。

ユーザモードでの使い方:

MODE WiZ -w

WIZからのWALLOPSメッセージの受信を無視します。

:Angel MODE Angel +i

Angelからのメッセージを自分自身で不可視にします

MODE WiZ -o

WiZを"deopping"します(IRCオペレータ状態を解除します)

このメッセージの単なる逆("MODE WiZ +o")は、OPERメッセージでバイパスされているので、クライアントには許可されていません。

[4.2.4 TOPICメッセージ]

Command: TOPIC
Parameters: <channel> [<topic>]

TOPICメッセージはチャンネルのトピックを見たり変更したりするために使用します。もし、<topic>がなかった場合、<channel>で指定されたチャンネルのトピックが返されます。<topic>のパラメータが送られた場合、チャンネルモードのパーミッションが許せばチャンネルのトピックが変更されます。

ニュメリックリプライ:
ERR_NEEDMOREPARAMS
ERR_NOTONCHANNEL
ERR_CHANOPRIVSNEEDED
RPL_NOTOPIC
RPL_TOPIC

例:
:Wiz TOPIC #test :New topic
クライアントWizがトピックを設定します。
TOPIC #test :another topic
#testに"another topic"というトピックを設定します。
TOPIC #test
#testのトピックをチェックします。

[4.2.5 NAMESメッセージ]

Command: NAMES
Parameters: [<channel>{,<channel>}]

NAMESメッセージを使うことによって、クライアントはそのクライアントが可視なチャンネル上のニックネームを全て一覧することができます。可視なチャンネルとはprivate(+p)やsecret(+s)などが設定されていない、実際にだれかいるチャンネルを言います。その指定が妥当なものなら、<channel>パラメータは情報を返すチャンネルを指定します。間違ったチャンネル名に対するエラーリプライはありません。<channel>パラメータが与えられなかった場合は、全てのチャンネルとその参加者のリストが帰ってきます。このリストの最後には、クライアントのいるチャンネル上もしくは可視なチャンネル上にいない可視なクライアントのリストが、「チャンネル」"*"として一覧表示されます。

ニュメリックリプライ:
RPL_NAMREPLY
RPL_ENDOFNAMES

例:

NAMES #twilight_zone,#42

チャンネルがあなたにとって可視ならば、#twilight_zoneと#42上の可視なクライアントを一覧表示します。

NAMES

全ての可視なチャンネルと可視なクライアントを一覧表示します。

[4.2.6 LISTメッセージ]

Command: LIST
Parameters: [<channel>{,<channel>} [<server>]]

LISTメッセージはチャンネルとそのチャンネルのトピックを一覧表示するのに使用します。<channel>パラメータが使われた場合、チャンネルのステータスを表示するだけです。プライベートチャンネルでは、クライアントがそのプライベートチャンネルにいるのでなければ、チャンネル"Prv"として、(トピックなしで)一覧表示されます。同様にシークレットチャンネルでは、クライアントが問題のチャンネルメンバーでなければ、全くリストにのりません。

ニュメリックリプライ:
ERR_NOSUCHSERVER
RPL_LISTSTART
RPL_LIST
RPL_LISTEND

例:
LIST
全チャンネルを一覧表示します。
LIST #twilight_zone,#42
#twilight_zoneと#42のチャンネルを一覧表示します。

[4.2.7 INVITE(招待)メッセージ]

Command: INVITE
Parameters: <nickname> <channel>

INVITEメッセージはチャンネルにクライアントを招待するために使用します。パラメータ<nickname>はターゲットのチャンネル<channel>に招かれる人のニックネームです。ターゲットとなるクライアントが招かれるチャンネルが存在しなくてはならないとか、有効なチャンネルでなければならないといった必要性はありません。クライアントをinvite-onlyなチャンネル(mode +i)に招くには、INVITEを送信するクライアントがそのチャンネル上でチャンネルオペレータの権限をもっていなくてはなりません。

ニュメリックリプライ:
ERR_NEEDMOREPARAMS
ERR_NOSUCHNICK
ERR_NOTONCHANNEL
ERR_USERONCHANNEL
ERR_CHANOPRIVSNEEDED
RPL_INVITING
RPL_AWAY

例:
:Angel INVITE Wiz #Dust
クライアントAngelはWizをチャンネル#Dustに招待しています。
INVITE Wiz #Twilight_Zone
Wizを#Twilight_Zoneに招待する命令です。

[4.2.8 KICK(追放)メッセージ]

Command: KICK
Parameters: <channel> <user> [<comment>]

KICKメッセージは強制的にチャンネルからクライアントを削除するときに使うことができます。これはチャンネルから「追い出だし(kicks them out)」(強制離脱)ます。チャンネルオペレータだけがチャンネルの外からきた他のクライアントを追い出すことができます。チャンネルからキックの犠牲者を削除する前に、KICKメッセージを受け取ったサーバは、これが有効かどうか(送信者が本当にチャンネルオペレータかどうか)チェックします。

ニュメリックリプライ:
ERR_NEEDMOREPARAMS
ERR_NOSUCHCHANNEL
ERR_BADCHANMASK
ERR_CHANOPRIVSNEEDED
ERR_NOTONCHANNEL

例:

KICK &Melbourne Matthew

&MelbourneからMatthewを追い出します

KICK #Finnish John :Speaking English

Johnを#Finnishから"Speaking English"を理由として(コメント)追い出します。

:WiZ KICK #Finnish John

JohnをチャンネルFinnishから削除するWiZからのKICKメッセージです。

注: KICKメッセージをパラメータを以下のように拡張することも可能です:

<channel>{,<channel>} <user>{,<user>} [<comment>]

[4.3 サーバへのクエリーとメッセージ]

サーバに対するクエリーメッセージ群はネットワークに接続しているサーバについての情報を返すように設計されています。接続している全てのサーバはこれらの要求に正確に応答しなくてはなりません。どんなものでもその不当な(または不足している)リプライはそのサーバが壊れたサインのと受け取られます。そして、状況が回復するまで、可能な限り速やかにそのサーバと接続を切断または接続を不可能にしなくてはいけません。
これらのクエリーでは、<server>として現れるパラメータの所に、いつでもニックネームまたはサーバまたはいく種類かのワイルドカードを使った名前がくる事を意味します。しかしながら、それぞれパラメータについても、たった一つのクエリーとリプライのセットが生成されます。

[4.3.1 VERSIONメッセージ]

Command: VERSION
Parameters: [<server>]

VERSIONメッセージはサーバプログラムのバージョンを要求するために使用します。オプショナルなパラメータ<server>はクライアントが直接接続していないサーバプロ

グラムのバージョンを請求する時に使用します。

ニュメリックリプライ:
ERR_NOSUCHSERVER
RPL_VERSION

例:

:Wiz VERSION *.se

"*.se"にマッチするサーバのバージョンをチェックするWizからのメッセージです。

VERSION tolsun.oulu.fi

サーバ"tolsun.oulu.fi"のバージョンをチェックします。

[4.3.2 STATS(状態)メッセージ]

Command: STATS
Parameters: [<query> [<server>]

STATSメッセージはある特定のサーバの統計を要求するのに使用します。 <server>パラメータが書かれていなかった場合、単にRPL_ENDOFSTATSが送信されてきます。サーバは下記の(ような)クエリーを供給できなくてはならないのですが、このメッセージの実装はサーバに高く依存します。
送信先のサーバ(パラメータ<server>で指定できます)に対して、クエリーとして1文字を指定します。中継するサーバはこれを無視し変更せずに転送します。 以下に述べる要求は現在のIRCで実装されているもので、サーバのセットアップ情報の大部分が供給されます。 これらはバージョンによって同じようにはサポートされていないかもしれません。クエリーの目的と現在使用されているリプライのフォーマットに矛盾しない、正しいリプライをサーバはSTATSメッセージに対してリプライしなくてはなりません。
現在サポートされている要求:

c

サーバが接続できるまたは接続を許可しているサーバの一覧を返します。

h

リーフ(ネットワークの末端)として扱われることが強制されるか、ハブ(ネットワークの中心)として動くことが許可されているか、という点についてのサーバの一覧を返します。

i

クライアントの接続することをサーバが許可しているホストのリストを返します。

k

サーバが禁止しているユーザ名/ホスト名の組み合わせのリストを返します。

l

サーバの接続のリストを返します。各接続の確立の持続時間、その接続の各方向からのオクテット単位のトラフィック量、メッセージが分かります。

m

サーバがサポートしているメッセージと(もし使用されているならば)それぞれの使
用回数のリストを返します。

o

通常のクライアントがIRCオペレータになれるホストのリストを返します。

y

サーバの初期設定ファイルからYクラス行を表示します。

u

サーバが稼働し続けている時間を示す文字列を返します。

ニュメリックリプライ:
ERR_NOSUCHSERVER
RPL_STATSCLINE
RPL_STATSNLINE
RPL_STATSILINE
RPL_STATSKLINE
RPL_STATSQLINE
RPL_STATSLLINE
RPL_STATSLINKINFO
RPL_STATSUPTIME
RPL_STATSCOMMANDS
RPL_STATSOLINE
RPL_STATSHLINE
RPL_ENDOFSTATS

例:

STATS m

接続しているサーバの使えるメッセージチェックします。

:Wiz STATS c eff.org

サーバeff.orgのC/N行の情報をWiZが要求しています。

[4.3.3 LINKメッセージ]

Command: LINKS
Parameters: [[<remote server>] <server mask>]

LINKSメッセージによって、サーバが要求に答えることで知ることのできる全サーバ
をクライアントは一覧表示する事ができます。返されるサーバのリストはマスクに
マッチするもののリストかマスクが指定されていなければ完全なリストです。
<server mask>に加えて<remote server>が指定されていた場合、LINKSメッセージは
その<remote server>にマッチした最初のサーバへ転送されます。そして、そのサー
バが要求に対して応答をします。

ニュメリックリプライ:
ERR_NOSUCHSERVER
RPL_LINKS
RPL_ENDOFLINKS

例:

LINKS *.au

"*.au"にマッチする名前を持つ全サーバのリスト

:WiZ LINKS *.bu.edu *.edu

WiZから"*.edu"にマッチする最初のサーバへの"*.bu.edu"にマッチするサーバのリストを要求するLINKSメッセージ

[4.3.4 TIMEメッセージ]

Command: TIME
Parameters: [<server>]

TIMEメッセージは指定したサーバのローカルタイム(地方時間)を要求するのに使用されます。<server>パラメータが指定されなかった場合、メッセージを扱うサーバが要求に対して応答を返します。

ニュメリックリプライ:
ERR_NOSUCHSERVER
RPL_TIME

例:

TIME tolsun.oulu.fi

サーバ"tolson.oulu.fi"の時間をチェックします。

:Angel TIME *.au

クライアントAngelが"*.edu"にマッチするサーバの時間をチェックします。

[4.3.5 CONNECT(接続)メッセージ]

Command: CONNECT
Parameters: <target server> [<port> [<remote server>]]

CONNECTメッセージは直ちに他のサーバとの新しい接続の確立を試る事をサーバに強制する時に使用します。CONNECTメッセージを実行する権限はIRCオペレータのみに与えられています。リモートサーバを指定した場合、サーバはCONNECTメッセージを<target server>の<port>に対して試みます。

ニュメリックリプライ:
ERR_NOSUCHSERVER
ERR_NOPRIVILEGES
ERR_NEEDMOREPARAMS

例:

CONNECT tolsun.oulu.fi

tolsun.oulu.fiとサーバの接続を試みます。

:WiZ CONNECT eff.org 6667 csd.bu.edu

WiZが送信したサーバcsd.bu.eduとサーバeff.org間がポート6667で接続するCONNECT
メッセージを試みます。

[4.3.6 TRACE(追跡)メッセージ]

Command: TRACE
Parameters: [<server>]

TRACEメッセージは指定されたサーバへの経路を探すのに使用されます。このメッセージを処理する各サーバはそのサーバの経路リンクを通過した事を示すリプライメッセージをTRACEメッセージの送信者に送信しなくてはいけません。こうして、経路追跡を使うことで得られる結果に似た一連のリプライが構成されます。リプライを送信した後、指定したサーバに到達するまでは、次のサーバにTRACEメッセージ送信しなくてはなりません。<server>パラメータが書き落とされていた場合、TRACEコマンドに対して、現在のサーバが直接接続しているサーバを示すメッセージを送信者に送り返さなくてはいけません。
パラメータ<server>によって指定されたサーバが実在する場合、そのサーバは自分に接続された全サーバ、及びクライアントをレポートすることが求められていますが、現在のクライアントについてはIRCオペレータのみ見ることが出来ます。パラメータ<server>によって指定された受信先がニックネームの場合、単にニックネームがリプライとして割り当てられます。

ニュメリックリプライ:
ERR_NOSUCHSERVER

TRACEメッセージを他のサーバに転送する場合、全ての中継サーバはTRACEメッセージがそこを通過し次にどこに転送されたかということ示すためにリプライ
RPL_TRACELINKを返さなくてはいけません。

RPL_TRACELINK

TRACEリプライは以下に述べるニュメリックリプライのいくつかの集まりで構成されます。

RPL_TRACECONNECTING
RPL_TRACEHANDSHAKE
RPL_TRACEOPERATOR
RPL_TRACEUSER
RPL_TRACESERVER
RPL_TRACESERVICE
RPL_TRACENEWTYPE
RPL_TRACECLASS

例:

TRACE *.oulu.fi

"*.oulu.fi"にマッチするサーバへの経路をTRACE(追跡)します。

:WiZ TRACE AngelDust

WiZのニックネーム AngelDusに対して送信したTRACEメッセージです。

[4.3.7 ADMIN(管理者)メッセージ]

Command: ADMIN
Parameters: [<server>]

ADMINメッセージは、指定したサーバまたは<server>が省略された場合は現在のサーバの管理者の名前を見つけるためにつかいます。各サーバはADMINメッセージを他のサーバに転送できなければなりません。

ニュメリックリプライ:
ERR_NOSUCHSERVER
RPL_ADMINME
RPL_ADMINLOC1
RPL_ADMINLOC2
RPL_ADMINEMAIL

例:

ADMIN tolsun.oulu.fi

tolsun.oulu.fiにADMINリプライを要求します。

:WiZ ADMIN *.edu

WiZからの最初に"*.edu"にマッチしたサーバへのADMINメッセージです。

[4.3.8 INFOメッセージ]

Command: INFO
Parameters: [<server>]

INFOメッセージはサーバについての以下のような情報を返すために使用します:
(a) サーバのバージョン
(b) いつコンパイルされたか
(c) パッチレベル
(d) いつ起動されたか
(e) 重要と思われるその他雑多な情報

ニュメリックリプライ:
ERR_NOSUCHSERVER
RPL_INFO
RPL_ENDOFINFO

例:

INFO csd.bu.edu

csd.bu.eduにINFOメッセージに対する応答を要求します。

:Avalon INFO *.fi

このINFOメッセージはAvalonからの最初に"*.fi"にマッチしたサーバにリプライを要求します。

INFO Angel

Angelの接続しているサーバにINFOメッセージに対するリプライを要求します。

[4.4 メッセージの送信]

IRCプロトコルの主な目的はクライアントが互いに通信する基礎を提供することです。PRIVMSGとNOTICEだけが、あるクライアントから他のクライアントへテキストメッセージを実際に送信可能なメッセージです。残りは信頼性や順序を保証したりすることを可能とするために使用されます。

[4.4.1 PRIVMSG(private message)メッセージ]

Command: PRIVMSG
Parameters: <receiver>{,<receiver>} <text to be sent>

PRIVMSGはクライアント間のプライベートメッセージを送信するのに使われます
<receiver>はメッセージを受け取る人のニックネームです。<receiver>はコンマで区切ることでニックネームやチャンネルのリストにすることもできます。
<receiver>パラメータはホストマスク(#mask)やサーバマスク($mask)を使用することもできます。どちらの場合も、サーバはマスクにマッチするサーバやホストにPRVMSGメッセージを配信するだけです。マスクは最低1つの"."を含んでいなくてはなりません。そして最後の"."の後にワイルドカードがあってはいけません。この制限は誰かがメッセージを"#*"や"$*"へ送信することを防ぐために存在します。これは、全てのクライアントに対するブロードキャストになってしまいます。経験的にこれらは正しく意図的に使用されることよりも誤用されることが多いからです。ワイルドカードは'*'と'?'の文字が使用できます。このPRIVMSGメッセージへの拡張はIRCオペレータのみが使用できます。

ニュメリックリプライ:
ERR_NORECIPIENT
ERR_NOTEXTTOSEND
ERR_CANNOTSENDTOCHAN
ERR_NOTOPLEVEL
ERR_WILDTOPLEVEL
ERR_TOOMANYTARGETS
ERR_NOSUCHNICK
RPL_AWAY

例:

:Angel PRIVMSG Wiz :Hello are you receiving this message ?

AngelからWizに対するメッセージです

PRIVMSG Angel :yes I'm receiving it !receiving it !'u>(768u+1n) .br

Angelへのメッセージです。

PRIVMSG jto@tolsun.oulu.fi :Hello !

サーバtolsun.oulu.fi上のユーザ名が"jto"というクライアントへのメッセージです

    PRIVMSG $*.fi :Server tolsun.oulu.fi rebooting.

"*.fi"にマッチする名前のサーバ上にいる全員へのメッセージです。

    PRIVMSG #*.edu :NSFNet is undergoing work, expect interruptions

"*.edu"にマッチする名前のホストから来た全クライアントへのメッセージです。

[4.4.2 NOTICEメッセージ]

      Command: NOTICE
      Parameters: <nickname> <text>

NOTICEメッセージはPRIVMSGと同じように使用します。NOTICEとPRIVMSGの間で異なる点は、NOTICEメッセージを受け取った場合、それに対してなにも反応して送信してはいけません。この規定はサーバに対しても適用されます。NOTICEメッセージを受け取ったクライアントにエラーリプライすら返信する必要してはいけません。この規定の目的は、メッセージに対して自動的になにかを送信するクライアントとの間のループを避けることです。これは典型的なオートマトン(クライアントはAIまたは人間が操作する会話的なプログラムです)によって使用されます。すなわち、他のオートマトンとのリプライのループが切りたいオートマトンが使用します。
リプライの詳細と例はPRIVMSGを参照のこと

[4.5 ユーザ情報の要求]

ユーザ情報の要求は特にクライアントやクライアントのグループ(チャンネル)について詳細を調べる関係したメッセージ群です。これらのメッセージにおいてワイルドカードを使用したとき、ワイルドカードがマッチした場合、「見える(visible)」クライアントの情報を返すだけです。クライアントの可視性はユーザモードと同一のチャンネルにいるかどうかという事の組み合わせに従って決定されます。

[4.5.1 WHOメッセージ]

      Command: WHO
      Parameters: [<name> [<o>]]

WHOメッセージはクライアントが<name>パラメータにマッチしたクライアントの情報のリストを返すクエリーを生成するために使用されます。<name>パラメータが欠如していた場合、全ての可視なクライアント(不可視(user mode +i)になっていないクライアントで、かつ、クエリーを送信したクライアントと共通のチャンネルにいないクライアント)のリストを表示します。"0"という<name>を使うか、全てのエントリに完全にマッチするワイルドカードを使うことで同じ結果を得ることができます。
WHOメッセージに渡された<name>は、channel<name>にマッチしない場合、クライアントのhost<name>,server<name>,real<name>,nick<name>に対してマッチします。
"o"パラメータが与えられた場合、与えられたネームマスクによる結果のうちIRCオペレータだけを返します。

 ニュメリックリプライ:
      ERR_NOSUCHSERVER
      RPL_WHOREPLY
      RPL_ENDOFWHO

例:

WHO *.fi

"*.fi"にマッチする全クライアントのリストを要求します。

WHO jto* o

"jto*"にマッチする全クライアントの内のIRCオペレータのリストを要求します。

[4.5.2 WHOISメッセージ]

Command: WHOIS
Parameters: [<server>] <nickmask>[,<nickmask>[,...]]

このWHOISメッセージはクライアント個別の情報を要求するために使用されます。
<nickmask>にマッチする(見る資格を持っている)各クライアントの状態の違いを示す
ニュメリックリプライによってこのメッセージに回答します。<nickmask>にワイル
ドカードが使われていない場合、可視なニックネームについての情報が表示されま
す。コンマ','でニックネームのリストを区切ることで複数指定することができま
す。
別の使い方では指定したサーバにクエリーを送信できます。他の情報は全体に行き
渡っているのですが、問題のユーザがどのくらいidle状態にあるかといったローカル
サーバ(すなわち、ユーザが直接に接続しているサーバ)にしか分からないことを知り
たい場合に便利です。

ニュメリックリプライ:
ERR_NOSUCHSERVER
ERR_NONICKNAMEGIVE
ERR_NOSUCHNICK
RPL_WHOISUSER
RPL_WHOISCHANNELS
RPL_WHOISSERVER
RPL_AWAY
RPL_WHOISOPERATOR
RPL_WHOISIDLE
RPL_ENDOFWHOIS

例:

WHOIS wiz

ニックネームがWiZのユーザ情報を可能な限り返します。

WHOIS eff.org trillian

サーバeff.orgにtrillianのユーザ情報を問い合わせます。

[4.5.3 WHOWASメッセージ]

Command: WHOWAS
Parameters: <nickname> [<count> [<server>]]

WHOWASメッセージはもう存在していない<nickname>の情報を問い合わせます。これ
はニックネームが変更されたり、IRCから去ったりした場合に使用します。このクエ
リーに対するリプライは、サーバがニックネームヒストリを使って<nickname>に該当

する見出しのエントリを探索します。(ここではワイルドカードマッチを使用しません。) ヒストリは後方検索され、最初に見つかったエントリを最初に返します。複数のエントリがあった場合、<count>個までのリプライが返されます。(<count>パラメータを指定しなかった場合は見つかるだけ全て返します。) <count>として非正数が渡された場合、見つかるだけ全てを返します。

ニュメリックリプライ:

ERR_NONICKNAMEGIVEN
ERR_WASNOSUCHNICK
RPL_WHOWASUSER
RPL_WHOISSERVER
RPL_ENDOFWHOWAS

例:

WHOWAS WiZ

ニックネームヒストリ中のニックネーム"WiZ"に関する全情報を返します。

WHOWAS Mermaid 9

最近9個までの"Mermaid"に関するニックネームヒストリのエントリを返します。

WHOWAS Trillian 1 *.edu

最初に"*.edu"にマッチしたサーバ上のヒストリで、最も新しい"Trillian"のエントリを返します。

[4.6 その他のメッセージ]

このカテゴリのメッセージはどのカテゴリにも属さないものですが、プロトコルの一部であり必要なものです。

[4.6.1 KILLメッセージ]

Command: KILL
Parameters: <nickname> <comment>

KILLメッセージはクライアント-サーバ間の接続を直接接続しているサーバが閉じるときに使われます。サーバが、正確なニックネームリストでだぶったエントリに遭遇したときに、この2つのエントリをKILLメッセージを使って削除します。これはIRCオペレータも使用することができます。
単に接続を短時間切断するだけのことなので、事実上自動的に再接続するアルゴリズムを持っているクライアントにはこのメッセージに利用価値はありません。クライアントは生成されたKILLメッセージを拒否することが出来ないので、問題の箇所に目星をつけてデータの流れを中断させ、大きな被害になることを食い止めるのに使うことができます。

ニックネームはいつでも世界中でユニークな事が要求されます。「二重性」が検知された時は(すなわち、2つのクライアントが同一のニックネームを登録しようとした時)いかなる時でも両者が消えるか、片方だけ再び出現することを期待して送信されます。
<comment>はKILLメッセージに対する実際の理由を反映していなくてはいけません。サーバが生成するKILLメッセージのコメントは2つの衝突したニックネームの所在に関する詳細な情報で構成されます。これを見た他のユーザが満足できる十分な理由をユーザが提供することは、それぞれにまかされています。偽のKILLメッセージの

送信者が自分自身の情報を隠すのを防ぐために、このコマンドを通過させるサーバは
仁の名前を'kill-path'と呼ばれるものを後ろに付け加えて、これを<comment>に含め
ます。

ニュメリックリプライ:
ERR_NOPRIVILEGES
ERR_NEEDMOREPARAMS
ERR_NOSUCHNICK
ERR_CANTKILLSERVER

KILL David (csd.bu.edu <- tolsun.oulu.fi)
ニックネームの衝突がcsd.bu.eduとtolson.oulu.fiの間で起こりました。

注:
KILLメッセージを使って他のクライアントをKILLする事はオペレータのみに許可され
ています。理想的にはIRCオペレータすらこれを行う必要はありません。これを使用
するものとしてはサーバだけが残ります。

[4.6.2 PINGメッセージ]

Command: PING
Parameters: <server1> [<server2>]

PINGメッセージはコネクションの他端のアクティブなクライアントの状態をテストす
るために使われます。他端の接続からアクティブなものが検知できない場合、PING
メッセージが一定間隔をおいて送信されます。接続が規定時間以内にPINGメッセー
ジに対して反応しなかった場合、その接続は閉じられます。
PINGメッセージを受け取ったクライアントは、まだ接続が生きていることを示すため
に、<server1>に対してできる限り速やかに適切なPONGメッセージを返信しなくては
いけません。サーバはPINGメッセージに対して返信はしません。しかし、PINGに対
するリプライから他端の接続からの接続が生きている事を信用します。<server2>パ
ラメータが指定された場合、PINGメッセージはそこに転送されます。

ニュメリックリプライ:
ERR_NOORIGIN
ERR_NOSUCHSERVER

例

PING tolsun.oulu.fi

別のサーバがまだ生きていること確かめるためにサーバが送信するPINGメッセージ

PING WiZ

PINGメッセージをニックネーム WiZに対して送信します。

[4.6.3 PONGメッセージ]

Command: PONG
Parameters: <daemon> [<daemon2>]

PONGメッセージはPINGメッセージに対するリプライです。パラメータ<daemon2>が指
定された場合、このメッセージは指定のデーモンに転送されます。<daemon>パラメ

ータはPINGメッセージに対してこのPONGメッセージを生成して返信を行うデーモンの名前です。

ニュメリックリプライ:
ERR_NOORIGIN
ERR_NOSUCHSERVER

例:
PONG csd.bu.edu tolsun.oulu.fi
csd.bu.eduからtolsun.oulu.fiへのPONGメッセージです

[4.6.4 ERRORメッセージ]

Command: ERROR
Parameters: <error message>

ERRORメッセージはIRCオペレータに、深刻なまたは致命的なエラーを報告するために、サーバが使用します。あるサーバから別のサーバに対して送信することもできます。しかし、通常、他端のクライアントからのものは受領されません。
ERRORメッセージはサーバ-サーバ間の接続で発生したエラーを報告するためだけに使用されます。ERRORメッセージは全サーバ(これは、直接接続しているオペレータ全員にERRORメッセージを伝えます)と直接接続している全オペレータに対して送信されます。あるサーバからERRORメッセージが届いた場合、サーバはERRORメッセージを他のサーバに転送しません。
サーバがIRCオペレータに受信したERRORメッセージを送信するときは、クライアントがこのエラーに対して反応しないように指示するために、<error message>をNOTICEメッセージにします。

ニュメリックリプライ:
存在しません。

例:

ERROR :Server *.fi already exists

このエラーが起きたサーバへのERRORメッセージです

NOTICE WiZ :ERROR from csd.bu.edu -- Server *.fi already exists

他端のサーバ上のクライアントWiZへ送る場合の、上記と同じようなERRORメッセージです。

[5. OPTIONALメッセージ]

このセクションではOPTIONALメッセージについて述べてゆきます。サーバはここのプロトコルで述べる実装をする必要はありません。このオプションがないときは、エラーリプライメッセージを生成するか、さもなければ、不明なメッセージのエラーとします。他のサーバがそのメッセージにリプライを送信するいる場合があるので、これを通過させなくてはいけません。(従って、要素を解析をする必要があります。) これらのリプライに割り当てられた番号はメッセージの説明の後に並べてあります。

[5.1 AWAYメッセージ]

Command: AWAY
Parameters: [message]

AWAY(チャンネルに対して送信されたものではなく)そのクライアントに直接配信されたPRIVMSGメッセージに対して自動的にリプライメッセージを返信するようにクライアントは指定することができます。クライアントに配信されたPRIVMSGメッセージに対してサーバが自動リプライを送信します。リプライをするサーバは送信したクライアントがつながっているサーバだけです。
パラメータを1つ指定した時(AWAYメッセージを設定する)とパラメータなしの時(AWAYメッセージを解除する)の両方の場合で使用できます。

ニュメリックリプライ:
RPL_UNAWAY
RPL_NOWAWAY

例:
AWAY :Gone to lunch. Back in 5
AWAYメッセージに"Gone to lunch. Back in 5"と設定する。
:WiZ AWAY
WiZのAWAYメッセージを解除します。

[5.2 REHASHメッセージ]

Command: REHASH
Parameters: None

REHASHメッセージは強制的にサーバに設定ファイルを読み込ませるためにIRCオペレータが使用します。

ニュメリックリプライ:
ERR_NOPRIVILEGES
RPL_REHASHING

例:

REHASH

IRCオペレータの権限をもつクライアントからサーバに対して設定ファイルを読み込む事を指示するメッセージです。

[5.3 RESTARTメッセージ]

Command: RESTART
Parameters: None

RESTARTメッセージは強制的にサーバを再起動するためにIRCオペレータが使用します。IRCオペレータとしてサーバに接続しているひとが、なにかの間違いでRESTARTメッセージを送信するリスクがあるので、このメッセージはオプションあつかいになっています。このメッセージは(最低でも)IRCサービスの中断を引き起こします。
RESTARTメッセージはいつでも、送信したクライアントが接続しているサーバに作用します。接続している他のサーバに転送されることはありません。

ニュメリックリプライ:

ERR_NOPRIVILEGES

例:
RESTART
パラメータは必要ありません

[5.4 SUMMON(召還)メッセージ]

Command: SUMMON
Parameters: <user> [<server>]

SUMMONメッセージはIRCサーバの走っているホストのユーザにIRCに参加するように呼びかけるメッセージを与えるために使用します。以下のような条件を満たす場合のみメッセージは送信されます:

(a) 送信先のサーバがSUMMONできる
(b) 目的とするユーザがログインしている
(c) サーバプロセスがそのユーザのtty(またはそれと同等のもの)に書き込みできる。

<server>パラメータが指定されなかった場合、クライアントがつないでいるサーバを送信先サーバと仮定して、<user>をSUMMON(召喚)することを試みます。
そのサーバでSUMMON(召喚)できなかった場合、ニュメリックリプライERR_SUMMONDISABLEDが返され、SUMMONメッセージは失敗します。

ニュメリックリプライ:
ERR_NORECIPIENT
ERR_FILEERROR
ERR_NOLOGIN
ERR_NOSUCHSERVER
RPL_SUMMONING

例:

SUMMON jto

サーバホスト上にいるユーザjtoを召還します。

SUMMON jto tolsun.oulu.fi

サーバ"tolsun.oulu.fi"の走っているホスト上にいるユーザjtoを召還します。

[5.5 USERSメッセージ]

Command: USERS
Parameters: [<server>]

USERSメッセージはそのサーバのユーザログインリストを、who(1)やrusers(1)、finger(1)のような形式で返します。セキュリティに関係する理由で、接続しているサーバ上でこのメッセージが使えない人がいる事もあります。使えない場合、この事を示すため、正しいニュメリックリプライを返さなくてはいけません。

ニュメリックリプライ
ERR_NOSUCHSERVER

ERR_FILEERROR
RPL_USERSSTART
RPL_USERS
RPL_NOUSERS
RPL_ENDOFUSERS
ERR_USERSDISABLED

使用されなくなったニュメリックリプライ:
ERR_USERSDISABLED

例:
USERS eff.org
サーバeff.org上のユーザログインリスト要求します。
:John USERS tolsun.oulu.fi
Johnからのサーバtolsun.oulu.fi上のロユーザグインリストの要求です。

[5.6 WALLOPS(operwall)メッセージ]

Command: WALLOPS
Parameters: 現在オンライン中の全IRCオペレータに送信するテキスト

現在オンライン中の全IRCオペレータに対してメッセージを送信します。ユーザメッセージとしてWALLOPSの実装をすると、(WALLとよく似た)多くの人にメッセージを送信する方法としてよく悪用されます。このため現在は、以下の例のように、WALLOPSの送信者としてはサーバのみを認めるようにWALLOPSを実装しなくてはなりません。

ニュメリックリプライ:
ERR_NEEDMOREPARAMS

例:
:csd.bu.edu WALLOPS :Connect '*.uiuc.edu 6667' from Joshua
JoshuaからのCONNECTメッセージをアナウンスするcsd.bu.eduからのWALLOPSメッセージです。

[5.7 USERHOSTメッセージ]

USERHOSTメッセージは最大5つのニックネームをスペース文字で区切ったリストをパラメータとしてとることができます。各ニックネームについて見つかった情報を返します。各リプライはスペースで区切ったリストとして返されます。

ニュメリックリプライ:
RPL_USERHOST
ERR_NEEDMOREPARAMS

例:
USERHOST Wiz Michael Marty p
ニックネームが"WiZ","Michael","Marty","p"の情報をUSERHOSTは要求します。

[5.8 ISONメッセージ]

Command: ISON
Parameters: <nickname>{<space><nickname>}

ISONメッセージはすばやく効果的にニックネームが現在IRC上にいるかどうかを得る方法を提供するために実装されていました。ISONはたった1つのパラメータをとりま

す。それはスペースで区切られたニックネームのリストです。 このリストにある各ニックネームがいた場合、サーバはリプライ文字列にそのニックネームを加えます。従って、空文字列(指定されたニックネームがいない時)から、パラメータ文字列の完全なコピー(全ていた時)まで、パラメータで指定したニックネームの部分集合がリプライの文字列として返されます。 チェックできるニックネームの数の制限は文字列の長さできまります。長すぎる場合はパラメータの文字列は512文字以内になるようにサーバが切り落とします。
ISONはローカルサーバがメッセージを発送したクライアントに対して実行します。遠方の他のサーバには転送しません。

ニュメリックリプライ:
RPL_ISON
ERR_NEEDMOREPARAMS

例:
ISON phone trillian WiZ jarlek Avalon Angel Monstah
7つのニックネームのISONを要求する例です。

[6. リプライ]

以下は、メッセージに応じて生成されるニュメリックリプライのリストです。 各々のニュメリックリプライにはその番号と名前とリプライ文字列が割り当てられています。

[6.1 エラーリプライ]

401 ERR_NOSUCHNICK
"<nickname> :No such nick/channel"
"<nickname> :そのようなnickやchannelはありません"

メッセージの与えたパラメータのニックネームが現在使用されていないことを示すために使用されます。

402 ERR_NOSUCHSERVE
"<server name> :No such server"
"<server name> :そのようserverはありません"

指定したサーバ名が存在しないことを示すために使用されます。

403 ERR_NOSUCHCHANNEL
"<channel name> :No such channel"
"<channel name> :そのようなchannelはありません"

与えたチャンネル名が不正であることを示すために使用されます。

404 ERR_CANNOTSENDTOCHAN
"<channel name> :Cannot send to channel"
"<channel name> :チャンネルに送れません"
(a)mode +nがセットされている場合、チャンネル上にいない
(b)mode +mがセットされている場合、チャンネル上でチャンネルオペレータではない(またはmode +vされていない)

以上のような条件で、クライアントがチャンネルにPRIVMSGメッセージを送信しようとすると、そのクライアントにこのメッセージが返信されます。

405 ERR_TOOMANYCHANNELS
"<channel name> :You have joined too many channels"

"<channel name> :あなたは多くのチャンネルにJOIN(参加)しすぎです"

許可されているチャンネル最大数にすでにJOINしているときに、もうひとつ他のチャンネルにJOINしようとしたクライアントにこのメッセージが返信されます。

406 ERR_WASNOSUCHNICK
"<nickname> :There was no such nickname"
"<nickname> :その様なニックネームは存在しません"

WHOWASメッセージは該当するニックネームの情報がヒストリ上にないことを示します。

407 ERR_TOOMANYTARGETS
"<target> :Duplicate recipients. No message delivered"
"<target> :2重に受信ました。メッセージは配信されません"

user@hostに該当するクライアントが複数存在している場合に、user@host形式であらわされた送信先へPRIVMSG/NOTICEを送信しようとしたクライアントに返されるエラーリプライです。

409 ERR_NOORIGIN
":No origin specified"
":送信者が指定されていません"

必要な送信者のパラメータを落としたPINGやPONGメッセージです。PINGやPONGメッセージは正しいプレフィックスなしでも働かなくてはならなりません。

411 ERR_NORECIPIENT
":No recipient given (<command>)"
":(<command>)の指定した受信先がありません"

412 ERR_NOTEXTTOSEND
":No text to send"
":送信すべきテキストがありません"

413 ERR_NOTOPLEVE
"<mask> :No toplevel domain specified"
"<mask> :最上位のドメインが指定されていません"

414 ERR_WILDTOPLEVEL
"<mask> :Wildcard in toplevel domain"
"<mask> :最上位のドメインでワイルドカードが使用されています"

PRIVMSGが412～414のリプライを何らかの理由によってメッセージが配信されなかったこと示すために返します。ERR_NOTOPLEVELとERR_WILDTOPLEVELは"PRIVMSG $<server>"や"PRIVMSG #<host>"の試行が不正に使用されたときに返されるエラーです

421 ERR_UNKNOWNCOMMAND
"<command> :Unknown command"
"<command> :不明なコマンドです"

送られたコマンド<command>がサーバにとって不明な事を示すためにクライアントに返されます。

422 ERR_NOMOTD
":MOTD File is missing"
":MOTDファイルがありません"

サーバがサーバのMOTDファイルを開くことができませんでした。

423 ERR_NOADMININFO
"<server> :No administrative info available"
"<server> :利用可能な管理情報がありません"

サーバが適切な情報を検索中にエラーがあったときADMINメッセージへの応答として返します。

424 ERR_FILEERROR
":File error doing <file op> on <file>"
":<file>上の<file op>の実行中にファイルエラーが起きました"

生成されたエラーメッセージはメッセージの処理中に起きた、ファイル操作の失敗を報告するために使用されます。

431 ERR_NONICKNAMEGIVEN
":No nickname given"
":ニックネームが指定されていません"

パラメータ<nickname>が必要なメッセージでパラメータが見つからない時に返します。

432 ERR_ERRONEUSNICKNAME
"<nickname> :Erroneous nickname"
"<nickname> :間違ったニックネームです"

定義された文字セットにない文字を含んだNICKメッセージを受け取った後に返します。

433 ERR_NICKNAMEINUSE
"<nickname> :Nickname is already in use"
"<nickname> :ニックネームはすでに使用中です"

NICKメッセージが現在すでに存在しているニックネームに変更しようとしたときに返します。

436 ERR_NICKCOLLISION
"<nickname> :Nickname collision KILL
"<nickname> :ニックネームの衝突が発生しKILLされました"

ニックネームの衝突(他のサーバにすでに存在しているニックネームを登録しようとした)を検知した時にサーバがクライアントに返します。

441 ERR_USERNOTINCHANNEL
"<nickname> <channel> :They aren't on that channel"
"<nickname> <channel> :彼らはそのチャンネル上にいません"

メッセージの送信先クライアントが指定のチャンネル上にいないこと示すためにサーバが返します。

442 ERR_NOTONCHANNE
"<channel> :You're not on that channel"
"<channel> :あなたはそのチャンネル上にいません"

クライアントがメンバーでないチャンネルに効果をおよぼすメッセージを実行しようとした時にサーバが返します。

443 ERR_USERONCHANNEL
"<user> <channel> :is already on channel"
"<user> <channel> :すでにチャンネル上にいます"

クライアントがすでにチャンネル<channel>上にいるユーザ<user>をチャンネルに招こうとした時に返されます。

444 ERR_NOLOGIN
"<user> :User not logged in"
"<user> :ユーザはログインしていません"

該当するユーザがログインしていないのでSUMMONメッセージを実行できなかった時に返されます。

445 ERR_SUMMONDISABLED
":SUMMON has been disabled"
":SUMMONは実行できません"

SUMMONメッセージに対する応答として返されます。この機能を実装していないサーバは返さなくてはなりません。

446 ERR_USERSDISABLED
":USERS has been disabled"
":USERSは実行できません"

USERSメッセージに対する応答として返されます。この機能を実装していないサーバは返さなくてはなりません。

451 ERR_NOTREGISTERED
":You have not registered"
":あなたは登録されていません"

サーバがメッセージの詳細を構文解析することを許可する前に、クライアントが登録しなくてはならないことを示すためにサーバが返します。

461 ERR_NEEDMOREPARAM
"<command> :Not enough parameters"
"<command> :パラメータが不足しています"

数々のメッセージにおいてパラメータが不足していることをクライアントに示すためにサーバが返します。

462 ERR_ALREADYREGISTRE
":You may not reregister"
":あなたは再登録できません"

登録の詳細(例えば、パスワードやUSERメッセージで得られるパスワードやユーザの詳細)の一部を変更しようとしている接続にサーバが返します。

463 ERR_NOPERMFORHOST
":Your host isn't among the privileged"
":あなたのホストは接続を許可されていません"

接続を試みようとしたホストからの接続が許可されるように設定されていないサーバに登録しようとしたクライアントに返されます。

464 ERR_PASSWDMISMATCH

":Password incorrect
":パスワードが間違っています"

パスワードが必要な時に、パスワードが指定されていない、または正しくないことが
原因で、接続を登録できなかったことを示すために返されます。

465 ERR_YOUREBANNEDCREEP
":You are banned from this server"
":あなたはこのサーバへの接続を禁止されています"

明示的に接続を拒否するように設定されているサーバに、拒否することが指定されて
いるものが登録／接続しようとすると返されます。

467 ERR_KEYSET
"<channel> :Channel key already set"
"<channel> :チャンネルkeyはすでに設定されています"

471 ERR_CHANNELISFULL
"<channel> :Cannot join channel (+l)"
"<channel> :(mode +lのために)チャンネルにJOINできません"

472 ERR_UNKNOWNMODE
"<char> :is unknown mode char to me"
"<char> :mode <char>が不明です。<char>の指定が間違っています"

473 ERR_INVITEONLYCHAN
"<channel> :Cannot join channel (+i)"
"<channel> :(mode +iのために)チャンネルにJOINできません"

474 ERR_BANNEDFROMCHAN
"<channel> :Cannot join channel (+b)"
"<channel> :(mode +bのために)チャンネルにJOINできません"

475 ERR_BADCHANNELKEY
"<channel> :Cannot join channel (+k)"
"<channel> :(mode +kのために)チャンネルにJOINできません"

481 ERR_NOPRIVILEGES
":Permission Denied- You're not an IRC operator"
":権限がありません- IRC operatorではありません"

行使するのにIRCオペレータの権限が必要なメッセージを実行しようして失敗したこ
とを示ために返すエラーです。

482 ERR_CHANOPRIVSNEEDED
"<channel> :You're not channel operator"
"<channel> :あなたはチャンネルオペレータではありません"

チャンネルオペレータの権限が必要なメッセージ(例えばMODEメッセージ)をチャンネ
ル上で指定されたチャンネルオペレータでないクライアントが実行しようとした時に
返すエラーです。

483 ERR_CANTKILLSERVER
":You cant kill a server!"
":サーバをKILLすることはできません!"

サーバ上でKILLメッセージは使えません。そしてこのエラーは直接クライアントに返
されます。

491 ERR_NOOPERHOST
":No O-lines for your host
":あなたのホストにはOラインがありません"

クライアントがOPERメッセージを送り、サーバがオペレータとしてクライアントのホストを接続することを許可しないように設定されている時、このエラーが返されます。

501 ERR_UMODEUNKNOWNFLAG
":Unknown MODE flag"
":不明なMODEフラグです"

MODEメッセージがニックネームと共に送信されていて、かつ送信されたMODEフラグが理解できないことを示すためにサーバが返します。

502 ERR_USERSDONTMATCH
":Cant change mode for other users"
":他のユーザのモードは変更できません"

自分自身以外の他のクライアントのユーザモードを見ようとしたまたは変えようとしたクライアントにエラーが返信されます。

[6.2 メッセージ応答]

300 RPL_NONE

この番号のリプライは使われません

302 RPL_USERHOST
":[<reply>{<space><reply>}]"

USERHOSTの使用するリプライの形式です。リプライの文字列は以下のような構成になります:

<reply> ::= <nick>['*'] '=' <'+'|'-'><hostname>

クライアントがIRCオペレータとして登録されているかどうかを'*'は示します。クライアントのAWAYメッセージが設定／解除されているか、それぞれ'-'/'+'で表しています。

303 RPL_ISON
":[<nickname> {<space><nickname>}]"

ISONの使用するリプライ形式です。

301 RPL_AWAY
"<nickname> :<away message>"

305 RPL_UNAWAY
":You are no longer marked as being away"

306 RPL_NOWAWAY
":You have been marked as being away"

これらのリプライはAWAYメッセージと共に使われます。AWAYを設定しているクライアントにPRIVMSGを送信したクライアントにRPL_AWAYが返信されます。クライアントが接続しているサーバがRPL_AWAYを送信します。クライアントがAWAYを解除/設定し

たときに、リプライRPL_UNAWAYとRPL_NOWAWAYが返信されます。

311 RPL_WHOISUSER
"<nickname> <user> <host> * :<real name>"

312 RPL_WHOISSERVER
"<nickname> <server> :<server info>"

313 RPL_WHOISOPERATOR
"<nickname> :is an IRC operator"

317 RPL_WHOISIDLE
"<nickname> <integer> :seconds idle"

318 RPL_ENDOFWHOIS
"<nickname> :End of /WHOIS list"

319 RPL_WHOISCHANNELS
"<nickname> :{[@|+]<channel><space>}

リプライ311～313及び317～319は全てWHOISメッセージに対する反応として生成されるリプライです。必要なパラメータがきちんと指定されていた場合、回答するサーバは 上記のニュメリックリプライの中からリプライを組み立るか、エラーリプライを返すかしなくてはなりません。RPL_WHOISUSERの中の'*'はリテラルな文字です。ワイルドカードではありません。各リプライのセットの内で、RPL_WHOISCHANNELSだけは複数回現れます。チャンネル名の後に続く'@'と'+'の文字はそれぞれクライアントがチャンネルオペレータまたはmoderated channel(mode +m)での発言権が与えられていることを示します。RPL_ENDOFWHOISリプライはWHOISメッセージを実行した後にその印として使われます。

314 RPL_WHOWASUSER
"<nickname> <user> <host> * :<real name>"

369 RPL_ENDOFWHOWAS
"<nickname> :End of WHOWAS"

WHOWASメッセージに対してリプライするとき、RPL_WHOWASUSER,RPL_WHOISSERVER,ERR_WASNOSUCHNICKのいずれかを利用して、指定されたリスト上の各ニックネームに対して、サーバはリプライを返さなくてはいけません。全てのリプライの束が処理し終わると、(一つしかリプライしなかったときや、エラーがあった場合でも)RPL_ENDOFWHOWASが送信されます。

321 RPL_LISTSTART
"Channel :Users Name"

322 RPL_LIST
"<channel> <# visible> :<topic>"

323 RPL_LISTEND
":End of /LIST"

リプライRPL_LISTSTART,RPL_LIST,RPL_LISTENDは、LISTメッセージに対するサーバの応答の、それぞれ、始まり、実際のリプライのデータ、終わりを表しています。返すべきチャンネルがなかった場合、単に始まり(RPL_LISTSTART)と終わり(RPL_LISTEND)が返送されるだけです。

324 RPL_CHANNELMODEIS
"<channel> <mode> <mode params>"

331 RPL_NOTOPIC
"<channel> :No topic is set"

332 RPL_TOPIC
"<channel> :<topic>"

TOPICメッセージがチャンネルトピックを決定するために送られたとき、2つのリプラ
イの内の一つが返信されます。トピックが設定されていた場合、RPL_TOPICが返信さ
れます。そうでない場合は、RPL_NOTOPICが返信されます。

341 RPL_INVITING
"<channel> <nickname>"

INVITEメッセージの試行が成功し、受信先のクライアントにメッセージが配信された
ことを示すためにサーバがこのリプライを返します。

342 RPL_SUMMONING
"<user> :Summoning user to IRC"

SUMMONメッセージの回答として、ユーザをSUMMON(召還)した事を示すためにサーバが
返します。

351 RPL_VERSION
"<version>.<debuglevel> <server> :<comments>"

サーバのバージョンの詳細を示すために、サーバが返します。 <version>は使ってい
るソフトウェアのバージョン(パッチレベル,リビジョンを含みます)です。そして、
<debuglevel>はサーバがデバッグモードで走っているかどうかを示します。
<comments>のフィールドにはバージョンのコメントか、より詳しいバージョンが入っ
ています。

352 RPL_WHOREPLY
"<channel> <user> <host> <server> <nickname> <HlG>[*][@|+] :<hopcount>
<real name>"

315 RPL_ENDOFWHO
"<name> :End of /WHO list"

RPL_WHOREPLYとRPL_ENDOFWHOの組はWHOメッセージの回答に使用されます。 WHOメッ
セージにマッチした場合だけに、RPL_WHOREPLYは返送されます。 WHOメッセージにパ
ラメータのリストがある場合、リストの各項の<name>に対してリプライが返信された
後にRPL_ENDOFWHOが返信されます。

353 RPL_NAMREPLY
"<channel> :[[@|+]<nick> [[@|+]<nick> [...]]]"

366 RPL_ENDOFNAME
"<channel> :End of /NAMES list"

NAMESメッセージにリプライするため、RPL_NAMREPLYとRPL_ENDOFNAMESのリプライの
組をクライアントにサーバが返信します。要求されたチャンネルが見つからない場
合、単にRPL_ENDOFNAMESを返すだけです。この例外として、NAMESメッセージがパラ
メータ無しで送られた時に、全ての可視チャンネルとそこにいるクライアントのリス
トが一連のRPL_NAMEREPLYリプライが返信され、最後にRPL_ENDOFNAMESが終わりの印
として返信されます。

364 RPL_LINKS

"<mask> <server> :<hopcount> <server info>"

365 RPL_ENDOFLINKS
"<mask> :End of /LINKS list"

LINKメッセージのリプライとして使われます。サーバはRPL_LINKSを返し、リストの
終わりの印としてRPL_ENDOFLINKSリプライを返します。

367 RPL_BANLIST
"<channel> <banid>"

368 RPL_ENDOFBANLIST
"<channel> :End of channel ban list"

サーバはRPL_BANLISTとRPL_ENDOFBANLISTリプライを使って、チャンネルに設定され
た'ban'のリストを返信します。 各RPL_BANLISTは有効なbanをそれぞれ示します。
banのリストが返信し終わった後(もしくは、存在しなかったら)、RPL_ENDOFBANLIST
が返信されます。

371 RPL_INFO
":<string>"

374 RPL_ENDOFINFO
":End of /INFO list"

INFOメッセージ対してサーバは全てのINFO(情報)を連続したRPL_INFOリプライとその
最後にリプライの終わりを示すRPL_ENDOFINFOリプライを返信する必要があります。

375 RPL_MOTDSTART
":- <server> Message of the day - "

372 RPL_MOTD
":- <text>"

376 RPL_ENDOFMOTD
":End of /MOTD command"

MOTDメッセージに対して反応しMOTDファイルを検索する場合、ファイル中で表示する
各行(各行は長くとも80文字まで)はRPL_MOTDの形式のリプライで返信されます。こ
れらRPL_MOTDは前はRPL_MOTDSTART、後ろはRPL_ENDOFMOTDとの間にはさまれなくては
いけません。

381 RPL_YOUREOPER
":You are now an IRC operator"

送信したOPERメッセージが成功しIRCオペレータの地位が授与されたクライアントに
RPL_YOUREOPERは返信されます。

382 RPL_REHASHING
"<config file> :Rehashing

REHASHオプションが使用され、IRCオペレータがREHASHメッセージを送信した場合、
RPL_REHASHINGがオペレータに返信されます。

391 RPL_TIME
"<server> :<string showing server's local time>"

サーバはTIMEメッセージ対して上記のRPL_TIMEの形式で返信しなくてはいけません。

時間の示す文字列はそこでの正確な日付と時間とを含んでいます。時間の文字列以
外は必要ありません。

392 RPL_USERSSTART
":UserID Terminal Host"

393 RPL_USERS
":%-8s %-9s %-8s"

394 RPL_ENDOFUSERS
":End of users"

395 RPL_NOUSERS
":Nobody logged in

サーバがUSERメッセージを受信した場合、リプライ
RPL_USERSTART,RPL_USERS,RPL_ENDOFUSERS,RPL_NOUSERSが使われます。
RPL_USERSSTARTは最初に送信されます。続いて連続してRPL_USERSが送信されるか、
RPL_NOUSERが一つ送信されます。最後にRPL_ENDOFUSERSが送信されます。

200 RPL_TRACELINK
"Link <version & debug level> <destination> <next server>"

201 RPL_TRACECONNECTING
"Try. <class> <server>"

202 RPL_TRACEHANDSHAKE
"H.S. <class> <server>"

203 RPL_TRACEUNKNOWN
"???? <class> [<client IP address in dot form>]"

204 RPL_TRACEOPERATOR
"Oper <class> <nick>"

205 RPL_TRACEUSER
"User <class> <nick>"

206 RPL_TRACESERVER
"Serv <class> <int>S <int>C <server> <nick!user|*!*>@<host|server>"

208 RPL_TRACENEWTYPE
"<newtype> 0 <client name>"

261 RPL_TRACELOG
"File <logfile> <debug level>"

全てのRPL_TRACE* はTRACEメッセージに対するサーバの応答として返されます。い
くつのリプライが返されるかは、TRACEメッセージとそれがIRCオペレータが送信した
のかそうでないのかによります。その量はあらかじめ決められてはいません。リプ
ライRPL_TRACEUNKNOWN,RPL_TRACECONNECTING,RPL_TRACEANDSHAKEは全て、確立されて
いない接続、不明な接続、接続がまだ試行中、'server handshake'の済んだプロセ
ス、等に対して使われます。RPL_TRACELINKはTRACEメッセージを受信したサーバが
返信します。そして、そのサーバは他のサーバに転送します。RPL_TRACELINKのリス
トはTRACEメッセージの通過した、実際にサーバがパスに沿って接続しているIRCネッ
トワークを反映するものとして返信されてきます。RPL_TRACENEWTYPEは、とりあえ
ず表示はされますが、どのカテゴリにも属さない接続に対して使われます。

211 RPL_STATSLINKINF
"<linkname> <sendq> <sent messages> <sent bytes> <received messages>
<received bytes> <time open>"

212 RPL_STATSCOMMANDS
"<command> <count>"

213 RPL_STATSCLINE
"C <host> * <name> <port> <class>"

214 RPL_STATSNLINE
"N <host> * <name> <port> <class>"

215 RPL_STATSILINE
"I <host> * <host> <port> <class>"

216 RPL_STATSKLINE
"K <host> * <username> <port> <class>"

218 RPL_STATSYLINE
"Y <class> <ping frequency> <connect frequency> <max sendq>"

219 RPL_ENDOFSTATS
"<stats letter> :End of /STATS report"

241 RPL_STATSLLINE
"L <hostmask> * <servername> <maxdepth>"

242 RPL_STATSUPTIME
":Server Up %d days %d:%02d:%02d"

243 RPL_STATSOLINE
"O <hostmask> * <name>"

244 RPL_STATSHLINE
"H <hostmask> * <servername>"

221 RPL_UMODEIS
"<user mode string>"

クライアントのモードについての要求にRPL_UMODEISを返信することで回答します。

251 RPL_LUSERCLIENT
":There are <integer> users and <integer> invisible on <integer>
servers"

252 RPL_LUSEROP
"<integer> :operator(s) online"

253 RPL_LUSERUNKNOWN
"<integer> :unknown connection(s)"

254 RPL_LUSERCHANNELS
"<integer> :channels formed"

255 RPL_LUSERME
":I have <integer> clients and <integer> servers

LUSERSメッセージが実行されると、サーバは
RPL_LUSERCLIENT,RPL_LUSEROP,RPL_USERUNKNOWN,RPL_LUSERCHANNELS,RPL_LUSERMEの
リプライからなるリプライのセットを返信します。リプライが行われると、サーバ
はRPL_LUSERCLIENTとRPL_LUSERMEを返信します。カウントがゼロでなかった場合は
他のリプライは単にそのまま返信されます。

256 RPL_ADMINME
"<server> :Administrative info"

257 RPL_ADMINLOC1
":<admin info>"

258 RPL_ADMINLOC2
":<admin info>"

259 RPL_ADMINEMAIL
":<admin info>"

ADMINメッセージに対してリプライする時、リプライRPL_ADMINMEから、
RPL_ADMINEMAILまでとそれぞれが提供するテキストメッセージをサーバは使用しなく
てはいけません。RPL_ADMINLOC1はどの国(country)のどの州(state)のどの町(city)
にいるかについて、続いて大学(university)や学部(department)の詳細
(RPL_ADMINLOC2)について、それから、最後にRPL_ADMINEMAILでサーバの管理者とコ
ンタクトをとるための情報(ここにはE-mailのアドレスを入ります)をサーバが返すこ
とを期待しています。

[6.3 予約番号]

以下のようなカテゴリに含まれるので、これらの番号は今まで記述されていません。

1. もはや使われていない
2. 将来の計画のために予約されている
3. 現在使われているが、現在のIRCサーバの性質上生成されない

209 RPL_TRACECLASS
217 RPL_STATSQLINE
231 RPL_SERVICEINFO
232 RPL_ENDOFSERVICES
233 RPL_SERVICE
234 RPL_SERVLIST
235 RPL_SERVLISTEND
316 RPL_WHOISCHANOP
361 RPL_KILLDON
362 RPL_CLOSING
363 RPL_CLOSEEND
373 RPL_INFOSTART
384 RPL_MYPORTI
466 ERR_YOUWILLBEBANNED
476 ERR_BADCHANMASK
492 ERR_NOSERVICEHOST

[7. クライアント／サーバ認証]

クライアントとサーバ双方は同じ認証レベルに属しています。いずれの場合も、サ
ーバとの接続を形成するために、ホスト名に対するIPアドレスの調査(およびこの反
対のチェック)が行われます。(もしパスワードがその接続に対して設定されていた
場合)どちらの接続もパスワード認証に属しています。パスワードチェックは一般的

にサーバにだけ使用されますが、パスワードチェックは全ての接続で可能です。
これに加えて、接続の認証をユーザ名を使用して行う方法が一般的になってきています。接続の他端のユーザ名を得る事は典型的なRFC1413で述べられているIDENTによるサーバ認証によって接続することを意味します。
パスワードを指定することなしに、ネットワーク接続の他の端点の信頼性を簡単に決定することはできません。IDENTの使用などの手段に加えて、パスワードの使用をサーバ間接続に関して強く推奨します。

[8. 現在の実装の詳細]

このプロトコルで述べる実装は"IRC server,version 2.8"現在のものです。これ以前のバージョンでは、多くのニュメリックリプライの代わりに、NOTICEメッセージを使って一部または全てのメッセージが実装されています。残念ながら、上位互換のために実装がこの公開されている文書と異なる実装がなされています。
主な相違点は:
* メッセージの終わりの印としてLFかCRのどちらかでも認識しています。(CR-LFが本来必要です)
このセクションの残は、大部分のサーバを実装しようとした場合の重要な問題を扱います。そしてクライアントにも同様にそのまま適用できる部分もあります。

[8.1 TCPネットワークプロトコル]

TCPが提供する信頼性あるネットワークプロトコルが会議の規模によく適しているので、IRCはTCPの上に実装されています。マルチキャストIPに置き換えられていきます。しかし、これはまだ幅広く利用できず、現時点ではサポートされていません。

[8.1.1 UNIXソケットのサポート]

UNIXのソケットではlisten/connectの操作が利用できます。現在の実装では、クライアントとサーバ両方の、UNIXのソケット上での接続でlistenとacceptするように構築されています。これらは'/'で始まるホスト名のところのソケットとして認識されます。
UNIXのソケットの接続についての情報が提供されたとき、実際のソケットの名前が特に指定されなければ、サーバはpathnameの中の実際のホスト名に置き換えられます。

[8.2 メッセージの構文解析]

クライアントとサーバに便利な「バッファリングのない(non-buffered)」ネットワークI/Oが提供されています。各接続にはそれぞれにプライベートな、最も新しく読み込み、パースの結果を保持している「入力バッファ(input buffer)」が割り当てられます。512バイトの大きさのバッファが1つの完全なメッセージを保持するために使われます。これは通常複数のメッセージを含んでいます。読み込みが行われた後、プライベートバッファは正しいメッセージかパースされます。そのバッファで一つのクライアントからの複数のメッセージを処理する場合、クライアントが消滅するかもしれないので気を付けなくてはなりません。

[8.3 メッセージの配信]

送信先のホストにデータが送信できない時や飽和したネットワークリンクが見つかる事はよくあることです。UNIXでは大抵これをTCP windowとその内部バッファを通して扱いますが、サーバは多くの場合、膨大な量の送信すべきデータを抱えています。(特に新しいサーバ-サーバリンクが形成された時です)そして、カーネルの提供する小さなバッファは、流入量に対して十分ではありません。この問題を軽減するため、「送信キュー(send queue)」が送信データのFIFOとして使われます。新しいサーバが接続する場合、低速なネットワーク接続の巨大なIRCネットワーク上では、よく送信キューが200Kbもの大きさに膨れることあります。接続の調査をする時、サーバはまず読み込み、全入力データをパースし、送信された各データはキューに格納されます。利用可能な全入力に対して実行されたら、キューにあったデータは吐き出

されます。これはwrite()システムコールの使用を減少させ、TCPがより大きなパケットを生成するのを助けます。

[8.4 接続の生涯]

接続が死んだり反応がなくなったりしたことが検知された場合、このような指定した一定時間に反応がなかった接続に対してサーバはPINGをしなくてはなりません。
接続が時間内に反応を返さなかった場合、この接続は適切な手順で閉じられます。
サーバプロセスをブロックするよりも、低速な接続を閉じた方がよいので、送信キューが最大値を超えた場合、接続が落ちます。

[8.5 サーバ-クライアント間の接続の確立]

IRCサーバとの接続をする場合、クライアントに(LUSERメッセージの通りの)現在のユーザ/サーバカウントだけではなく、(存在するならば)MOTDを送信します。適切な導入メッセージだけではなく、名前、バージョンをクライアントに明確なメッセージで指定しなくてはいけません。
これを処理してから、サーバは新しいクライアントのニックネームやそれについての情報を送信しなくてはなりません(USERメッセージによる)。それからサーバとして認識されます。(DNS認証サービスによる) サーバはUSERに続いてまずNICKの情報を送信します。

[8.6 サーバ-サーバ間の接続の確立]

問題が発生する可能性のある場所が多いので(最悪の場合競合状態が起こります)、サーバ-サーバ間の接続の確立する過程には危険がともないます。サーバが正しく認識されたPASS/SERVERの組に続いて接続が受け付けられた後、全状態情報(後述)と共に、その接続に関するPASS/SERVERの情報をリプライします。
サーバとの接続を受け付ける(accept)前に、初期化中の場合はPASS/SERVERの組を受信して、サーバの応答が正しく認証できるかもチェックします。

[8.6.1 サーバによる接続の状態情報の交換]

サーバ間で交換される状態情報のオーダーは必要不可欠なものです。
そのオーダーは以下に述べるようになります:

* 現在、既知の全サーバ
* 現在、既知の全ユーザ情報
* 現在、既知の全チャンネル情報

サーバに関する情報は 特別なSERVERメッセージを経由して送信されます。ユーザ情報は各NICK/USER/MODE/JOINメッセージで、チャンネル情報はMODEメッセージで送信されます。
注:
TOPICメッセージは古いトピック情報を上書きするだけですので、チャンネルトピックはここでは交換されません。接続の両端でトピックが交換されるだけです。
まず、サーバについての状態情報が転送されると、ニックネームの衝突が発生する前に、すでに存在しているサーバとの衝突が起こります。IRCネットワークは非循環的なグラフとして存在しているため、ネットワークが既に別の所で再結合されているかもしれません。この時、衝突の起こったところがネットを分断すべき場所になります。

[8.7 サーバ-クライアント間の接続の切断]

クライアントの接続が閉じられた時、クライアントとの接続を閉じるサーバが、クライアントのために、QUITメッセージが生成します。これ以外のメッセージが生成／使用されることはありません。

[8.8 サーバ-サーバ間の接続の切断]

サーバ-サーバ間の接続が、間接的にSQUITを用いて閉じられるか、「自然に」閉じてしまった場合、残りのIRCネットワークは閉鎖が検知されたサーバについての情報を更新しなくてはなりません。。サーバは(その接続の背後に広がっていた各サーバそれぞれに対応した)SQUITメッセージのリストと、(その接続の背後に広がっていた各クライアントに対応した)QUITメッセージのリストを送信します。

[8.9 ニックネーム変更の探知]

全IRCサーバは最新のニックネームの変更のヒストリを保持しなくてはなりません。
これはメッセージの操作によってニックネーム変更の競合状態が発生した時に接触を保つチャンスをサーバに与えるために必要です。
ニックネームの変更を追跡しなくてはならないメッセージは:

* KILL (ニックネームが削除される)
* MODE (+/- o,v)
* KICK (ニックネームが追放される)

その他のメッセージはニックネームの変更をチェックする必要はありません。
上記のような場合、サーバはまず最初にニックネームが存在するかどうか、そして(あるならば)誰のものかチェックしなくてはなりません。これで競合状態の危険性が減少しますが、それでもなお不適切なクライアントの影響でサーバで競合状態が発生することがあります。上記のメッセージに対して変更の追跡が行われた時、時間の範囲を設けて古すぎるエントリを無視するようにする事を推奨します。
正常なヒストリでは、サーバは以前のニックネームを保持することができます。これは全てのクライアントについて変更されたかどうかを知るためです。このサイズの制限は(例えばメモリのような)他の原因によります。

[8.10 クライアントのデータ流入管理]

IRCサーバ間で接続する大きなネットワークでは、ネットワークに接続している単一のクライアントが非常に簡単に絶え間のないメッセージを送信することができてしまいます。その結果、ネットワークで洪水が発生するだけでなく、他のクライアントに提供されるサービスの水準が低下します。洪水の「被害者」の保護をすることより、洪水を防ぐようにサーバは書かれています。そしてサーバ以外の全てのクライアントに対してこれが適用されます。
現在のアルゴリズムは以下のようになっています:

(1) クライアントの「メッセージタイマ」が現在の時間より小さいかどうかチェックします。(小さくなれば等しくします)
(2) クライアントから送信されてきた現在のデータを読み込みます。
(3) 現在のタイマの時刻が10秒以下の時は、現在のメッセージをパースしてます。

そしてクライアントに対してメッセージ1つに対して2秒のペナルティーを課します。
このようにして本質的な意味では不利な影響は及ぼさずに、クライアントは2秒毎に1メッセージを出すことができます。

[8.11 ノンブロッキング検索]

リアルタイム環境では、全てのクライアントが公平にサービスを受けることができるようにサーバプロセスの待ち時間をできるだけ短くする事が基本です。明らかにこれは全ネットワークのread/writeの操作についてノンブロッキングI/Oでなされる必要があります。通常のサーバ接続では、これは難しいことではありません。しかし、ブロックしなくてはならないサポート操作(例えばディスクの読み書き)も他にあ

ります。可能な限り、このような操作は短い時間で実行しなくてはいけません。

[8.11.1 ホスト名の(DNSのよる)検索]

バークレーの標準ライブラリを使用はタイムアウトのリプライが起きた場合、大きな遅延が起きます。これを避けるため、DNSルーチンのセットを分けて、段階毎にノンブロッキングI/O操作を行い、サーバのメインI/Oループの中で調査するように書かれています。

[8.11.2 ユーザ名の(IDENTによる)検索]

他のプログラムに組み込んで使用できるいくつかのIDENTライブラリがありますが、これらは同期方法に影響しいくつかの問題が起こります。そしてその結果、周期的な遅延が発生します。解決策は、ノンブロッキングI/Oとして使えるようサーバの残りと協調する形で書くことです。

[8.12 設定ファイル]

柔軟性のあるセットアップの方法とサーバの運用を提供するため、設定ファイルに以下に述べるサーバへの指示項目が含まれる事が推奨されます。

* クライアントの接続を受け付けるホスト
* サーバとして接続する事を許すホスト
* 接続するホスト(受動／能動、両方)
* サーバがどこにあるかについての情報(大学、区、市、県、州、国など /91-22,)
* 誰が サーバを管理していてるか。また、その人と連絡とるためのE-mailアドレス。
* IRCオペレータ限定のメッセージの利用権を要求するクライアントのための、ホスト名とpassword

ホスト名を指定では、ドメイン名と点'.'(dot)を使った表記法(例えば127.0.0.1)両方が受け付けられるようにすべきです。全ての送信／受信の接続の使用／受け付けに対してパスワードを指定することを可能にしなくてはいけません。(ただし、送信用の接続は他のサーバがこれを行います)
上記のリストはサーバが他のサーバと接続するためのに最小限必要なことです。
その他、役に立つかもしれない事項は:

* 他のサーバも紹介するサーバを指定する
* サーバ枝の深さの制限
* クライアントが接続していられる時間

[8.12.1 接続を許可するクライアント]

サーバは「アクセスコントロールリスト(access control list)」のようなものを使用すべきでしょう。それは起動時に読み込まれ、どのホスト、クライアントがそのサーバとの接続を利用できるか決定するために使用されます。「拒否(deny)」と「許可(allow)」の両方がホストアクセスコントロールの柔軟性を提供するために実装されるべきです。

[8.12.2 オペレータ]

破壊的な人物にオペレータの権限を許可することは、その与えた力によって、全IRCネットワークの安寧に悲惨な結果をもたらすことになるでしょう。それ故、このような権限の取得はあまり簡単にすべきではありません。現在、それら一方は簡単に推測できますが、2つのpasswordsが必要です。設定ファイル内のOPER passwordのストアは、簡単に盗みとられないようにcryptフォーマット(すなわちUNIXのcrypt(3)を使用します)で暗号化してハードコーディングする事が望ましいでしょう。

[8.12.3 接続を許可するサーバ]

サーバ間の接続 不正な接続はIRCの有用性に大きな衝撃を与えます。それゆえ、各サーバは、相手が自分の接続を許可していて、自分も相手の接続を許可しているサーバのリストを保持しているべきです。サーバが気まぐれなホストがサーバとして接続することを許すような状況を作ってはいけません。サーバが接続できるかできないということに加えて、設定ファイルにはpasswordや、そのリンクの他の特性などもストアすべきでしょう。

[8.12.4 経由地の管理]

ADMINメッセージ(セクション 4.3.7を参照のこと)に対する綿密で正確なリプライを提供できるように、サーバが設定の中で詳細な事項を取得できるようにしなくてはなりません。

[8.13 チャンネルのメンバー]

現在のサーバではローカルクライアントは最大10の違うチャンネルにJOINを登録できます。ローカルでないユーザの接続には制約をかすことはないので、サーバは(もちろん)全てのチャンネルの基本的なメンバーとして恒常的に加わっています。(つまりはサーバは全てのチャンネルのメンバーです。)

[9. 現在の問題]

このプロトコルにはいくつかの問題が認識されています。ncigt h evr#Iイプロトコkoテキ全て解決されることを願っています。現在、これらの解決策を探す作業が進行中です。

[9.1 スケーラビリティ]

大きなエリアで使った時、このプロトコルは十分に釣り合いがとれていない事が幅広く認識されています。主な問題点は、他のサーバ、クライアント、そして、これらの変更時に更新するための情報を全サーバが知っていなくてはならないという事です。パスの長さが各2点間で最小になるようにサーバの数をできるだけ減らし、また、、できるだけ強い枝からなる伸縮自在な木になるようにするため、いくつかのサーバの下で管理することが望ましいでしょう。

[9.2 ラベル]

現在IRCプロトコルは3タイプのラベルがあります: ニックネーム,channel name,server nameです 各3タイプはそれぞれの別々の領域(名前空間)に属しており、同じ領域で重複することはありません。
現在、クライアントは3タイプのラベルどれでも選択して使用するすことが可能です。そしてこのために衝突が発生する可能性があります。循環的なな木構造を許すさないのはもちろん、チャンネルやニックネームが衝突しないしないようなユニークな名前をつけるプランで再構築する必要があることが広く認識されています。

[9.2.1 ニックネーム]

IRC上のニックネームのアイデアは、クライアントがチャンネルの外の人と話すときに、非常に便利なものです。ニックネーム空間には数の限りがあります。しかし、同じニックネームを使用したい人が複数いることは珍しいことではありません。このプロトコルで2人の人が同一のニックネームを選んだ場合、一方が成功しないか、両方がKILLメッセージ(セクション4.6.1参照)によって削除されます。

[9.2.2 channel]

現在チャンネルレイアウトでは全サーバが全チャンネルと、チャンネルにいる人、チャンネルの性質を知らなくてはなりません。大きさと同時にプライバシーにも問題があります。ニックネームの衝突を解決したときのように排他的に解決されるのではなく、チャンネルの衝突は包括的な方法で解決します。(新たにチャンネルを作った人、両方ともが同じチャンネルのメンバーであるとみなします。)

[9.2.3 サーバ]

サーバの数は通常、クライアントやチャンネルの数くらべて相対的に少ないですが、現在、サーバは2つの方法で全体のなかから指定できます。それぞれ個別に指定するか、マスクでくるんで指定する方法です。

[9.3 アルゴリズム]

サーバのコードの中の複数箇所で、(例えば、クライアントのセットのリストからチャンネルをチェックするような所で)O(N^2)のアルゴリズムを使用することを避けることができません。
現在のサーバのバージョンでは、各サーバは隣接しているサーバが正常であることを想定していて、堅牢にチェックされたデータベースがありません。接続したサーバがバグを持っていたり、実在するネットワークと矛盾することを紹介しようとした場合、このことは大きな問題につながります。
現在、内部と外部のラベルのユニークさの欠如から、競合状態が多数存在してます。これら競合状態は一般的に、メッセージが転送され、IRCネットワークに波及するのにかかる時間が原因で生じます ユニークなラベルの変更でさえ、チャンネル関係のメッセージが分裂する問題があります。

[10. 現在のサポートと入手先]

IRCに関係する議論の行われているメーリングリスト:
　将来のプロトコルについて: ircd-three-request@eff.org
　全般的な話題:operlist-request@eff.org
ソフトウェアの実装:
　cs.bu.edu:/irc
　nic.funet.fi:/pub/irc
　coombs.anu.edu.au:/pub/irc
ニュースグループ:
　alt.irc

[11. セキュリティに対する配慮]

セキュリティについては、セクション 4.1, 4.1.1, 4.1.3, 5.5, 7で述べられています。

[12. 筆者の宛先]

Jarkko Oikarinen
Tuirantie 17 as 9
90500 OULU
FINLAND

Email: jto@tolsun.oulu.fi

Darren Reed
4 Pateman Street
Watsonia, Victoria 3087
Australia

Email: avalon@coombs.anu.edu.au

[13. 日本語訳にあたって]

日本語訳 12/20,1996改訂版

[13.1 謝辞]

東北大学理学部の福田竜生さん:
本文全体に渡り、非常に多くの、誤入力、誤訳さらには適切な修正、訳案を頂きました。福田さんの協力にこの場をかりて感謝の意を表したいと思います。

[13.2 日本語訳について]

日本語訳にあたって、できる限り自然な日本語になるようつとめました。一部訳しにくい(もしくは訳者の英語力の不足のため理解ができない)文章はかなり意訳してあるところもあります。不明瞭な所は適宜RFC1459原文の該当箇所を参照して下さい。意味の通じないところ、訂正すべき箇所、適切な訳案がございましたら、E-mailにてご連絡くだされば幸いです。

[13.3 訳者の連絡先]

早稲田大学理工学部電子通信学科 野々垣直浩
Email:g95g0819@mn.waseda.ac.jp

重ねてみなさまのご協力、よろしくお願いいたします。

[14. HTML化について]

野々垣さんの日本語訳をもとに、kabehanaがHTML化しました。(1997/09/04)

公開場所(過去1): http://www.mars.dti.ne.jp/%7Ewall/etc/rfc1459j.html
(1998/03/24まで)

公開場所(過去2): http://www.fugen.org/%7Ekabehana/etc/rfc1459j.html
(1998/03/24～2000/01/27まで)

公開場所(現在): http://kabehana.hoops.ne.jp/etc/rfc1459j.html (2000/01/27以降)

メイルアドレス変更(dti->fugen) (1999/06/29)
---end of file---