RFC 2813 - Internet Relay Chat: Server Protocol 日本語訳

原文URL : https://datatracker.ietf.org/doc/html/rfc2813
タイトル : **RFC 2813 - インターネットリレーチャット：サーバープロトコル**
翻訳編集 : 自動生成、ST : **Informational**


```
Network Working Group                                          C. Kalt
Request for Comments: 2813                                  April 2000
Updates: 1459
Category: Informational
```


Internet Relay Chat: Server Protocol

インターネットリレーチャット：サーバープロトコル

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

本文書の位置付け

このメモは、インターネットコミュニティに情報を提供します。いかなる種類のインターネット標準を指定しません。このメモの配布は無制限です。

Copyright Notice

著作権表示

Abstract

While based on the client-server model, the IRC (Internet Relay Chat) protocol allows servers to connect to each other effectively forming a network.

This document defines the protocol used by servers to talk to each other. It was originally a superset of the client protocol but has evolved differently.

First formally documented in May 1993 as part of RFC 1459 [IRC], most of the changes brought since then can be found in this document as development was focused on making the protocol scale better. Better scalability has allowed existing world-wide networks to keep growing and reach sizes which defy the old specification.

概要

クライアントサーバーモデルに基づいている間、IRC（インターネットリレーチャット）プロトコルにより、サーバーは互いに効果的にネットワークを形成することができます。

このドキュメントでは、サーバーが互いに通信するために使用されるプロトコルを定義します。もともとはクライアントプロトコルのスーパーセットでしたが、異なる進化を進めています。

1993年5月にRFC 1459 [IRC]の一部として最初に正式に文書化されました。それ以降の変更のほとんどは、プロトコルスケールの改善に焦点を当てているため、このドキュメントにもたらされました。より良いスケーラビリティにより、既存の世界的なネットワークは、古い仕様に逆らうサイズを拡大し続けることができます。

Table of Contents

目次

1. Introduction

This document is intended for people working on implementing an IRC server but will also be useful to anyone implementing an IRC service.

Servers provide the three basic services required for realtime conferencing defined by the "Internet Relay Chat: Architecture" [IRC-ARCH]: client locator (via the client protocol [IRC-CLIENT]), message relaying (via the server protocol defined in this document) and channel hosting and management (following specific rules [IRC-CHAN]).

1. はじめに

このドキュメントは、IRCサーバーの実装に取り組んでいる人々を対象としていますが、IRCサービスを実装している人にも役立ちます。

サーバーは、「インターネットリレーチャット：アーキテクチャ」[IRC-ARCH]：クライアントロケーター（クライアントプロトコル[IRC-Client]を介して）で定義されたリアルタイム会議に必要な3つの基本サービスを提供し、メッセージリレー（これで定義されたサーバープロトコルを介して）ドキュメント）およびチャネルホスティングと管理（特定のルールに従う[IRC-chan]）。

2. Global database

Although the IRC Protocol defines a fairly distributed model, each server maintains a "global state database" about the whole IRC network. This database is, in theory, identical on all servers.

2. グローバルデータベース

IRCプロトコルはかなり分散されたモデルを定義していま
各サーバーはIRCネットワーク全体に関する「グローバル状態データベース」を維持しています。このデータベースは、理論的にはすべてのサーバーで同じです。

2.1 Servers

Servers are uniquely identified by their name which has a maximum length of sixty three (63) characters. See the protocol grammar rules (section 3.3.1) for what may and may not be used in a server name.

Each server is typically known by all other servers, however it is possible to define a "hostmask" to group servers together according to their name. Inside the hostmasked area, all the servers have a name which matches the hostmask, and any other server with a name matching the hostmask SHALL NOT be connected to the IRC network outside the hostmasked area. Servers which are outside the area have no knowledge of the individual servers present inside the area, instead they are presented with a virtual server which has the hostmask for name.

2.1 サーバー

サーバーは、最大63文字（63）の文字を持つ名前で一意に識別されます。サーバー名で使用される可能性のあるもの、および使用されないものについては、プロトコル文法ルール（セクション3.3.1）を参照してください。

通常、各サーバーは他のすべてのサーバーで既知ですが、名前に応じてサーバーをグループ化するために「ホストマスク」を定義することができます。ホストマスク領域内には、すべてのサーバーにはホストマスクに一致する名前があり、ホストマスクに一致する名前を持つ他のサーバーは、ホストマスク領域の外側のIRCネットワークに接続してはなりません。エリアの外側にあるサーバーには、エリア内に存在する個々のサーバーの知識がなく、代わりに名前のホストマスクがある仮想サーバーが表示されます。

2.2 Clients

For each client, all servers MUST have the following

2.2 クライアント

各クライアントについて、すべてのサーバーには次の情報が必

information: a netwide unique identifier (whose format depends on the type of client) and the server to which the client is connected.

要です。ネットワイドの一意の識別子（フォーマットはクライアントのタイプに依存します）とクライアントが接続されているサーバーです。

2.2.1 Users

Each user is distinguished from other users by a unique nickname having a maximum length of nine (9) characters. See the protocol grammar rules (section 3.3.1) for what may and may not be used in a nickname. In addition to the nickname, all servers MUST have the following information about all users: the name of the host that the user is running on, the username of the user on that host, and the server to which the client is connected.

2.2.1 ユーザー

各ユーザーは、最大長さ9文字の一意のニックネームで他のユーザーと区別されます。ニックネームで使用される可能性のあるものとできないものについては、プロトコル文法規則（セクション3.3.1）を参照してください。ニックネームに加えて、すべてのサーバーには、すべてのユーザーに関する次の情報が必要です。ユーザーが実行しているホストの名前、そのホストのユーザーのユーザー名、およびクライアントが接続されているサーバーです。

2.2.2 Services

Each service is distinguished from other services by a service name composed of a nickname and a server name. The nickname has a maximum length of nine (9) characters. See the protocol grammar rules (section 3.3.1) for what may and may not be used in a nickname. The server name used to compose the service name is the name of the server to which the service is connected. In addition to this service name all servers MUST know the service type.

Services differ from users by the format of their identifier, but more importantly services and users don't have the same type of access to the server: services can request part or all of the global state information that a server maintains, but have a more restricted set of commands available to them (See "IRC Client Protocol" [IRC-CLIENT] for details on which) and are not allowed to join channels. Finally services are not usually subject to the "Flood control" mechanism described in section 5.8.

2.2.2 サービス

各サービスは、ニックネームとサーバー名で構成されるサービス名によって、他のサービスと区別されます。ニックネームの最大長は9文字です。ニックネームで使用される可能性のあるものとできないものについては、プロトコル文法規則（セクション3.3.1）を参照してください。サービス名を作成するために使用されるサーバー名は、サービスが接続されているサーバーの名前です。このサービス名に加えて、すべてのサーバーはサービスタイプを知っている必要があります。

サービスは識別子の形式によってユーザーとは異なりますが、より重要なことには、サービスとユーザーはサーバーへのアクセスと同じタイプのアクセスを持っていません。サービスは、サーバーが維持するグローバルな状態情報の一部またはすべてを要求できますが、より多くのグローバルな状態情報を要求できますが、利用可能なコマンドの制限付きセット（詳細については、「IRCクライアントプロトコル」[IRC-Client]を参照）を参照してください。最後に、サービスは通常、セクション5.8で説明されている「洪水制御」メカニズムの対象ではありません。

2.3 Channels

Alike services, channels have a scope [IRC-CHAN] and are not necessarily known to all servers. When a channel existence is known to a server, the server MUST keep track of the channel members, as well as the channel modes.

2.3 チャネル

同様に、チャンネルにはスコープ[IRC-chan]があり、すべてのサーバーに必ずしも知られているわけではありません。チャネルの存在がサーバーに存在する場合、サーバーはチャネルメンバーとチャネルモードを追跡する必要があります。

3. The IRC Server Specification

3. IRCサーバー仕様

3.1 Overview

The protocol as described herein is for use with server to server connections. For client to server connections, see the IRC Client Protocol specification.

There are, however, more restrictions on client connections (which are considered to be untrustworthy) than on server connections.

3.1 概要

本明細書に記載されているプロトコルは、サーバーからサーバーへの接続で使用するためのものです。クライアントからサーバーへの接続については、IRCクライアントプロトコルの仕様を参照してください。

ただし、サーバー接続よりもクライアント接続（信頼できないと考えられている）には、より多くの制限があります。

3.2 Character codes

No specific character set is specified. The protocol is based on a a set of codes which are composed of eight (8) bits, making up an octet. Each message may be composed of any number of these octets; however, some octet values are used for control codes which act as message delimiters.

Regardless of being an 8-bit protocol, the delimiters and

3.2 文字コード

特定の文字セットは指定されていません。プロトコルは、8つのビットで構成されるコードのセットに基づいており、オクテットを構成しています。各メッセージは、これらのオクテットの任意の数で構成されている場合があります。ただし、いくつかのオクテット値は、メッセージデリミターとして機能する制御コードに使用されます。

8ビットのプロトコルであることに関係なく、デリミターとキー

keywords are such that protocol is mostly usable from US-ASCII terminal and a telnet connection.

Because of IRC's Scandinavian origin, the characters {}|^ are considered to be the lower case equivalents of the characters []\~, respectively. This is a critical issue when determining the equivalence of two nicknames, or channel names.

ワードは、プロトコルが米国ASCII端末とTelnet接続からほとんど使用可能であるようなものです。

IRCのスカンジナビア起源のため、キャラクター{} |^は、それぞれ文字[] \〜の小文字と同等であると考えられています。これは、2つのニックネームまたはチャネル名の等価性を決定する場合の重要な問題です。

3.3 Messages

Servers and clients send each other messages which may or may not generate a reply. Most communication between servers do not generate any reply, as servers mostly perform routing tasks for the clients.

Each IRC message may consist of up to three main parts: the prefix (OPTIONAL), the command, and the command parameters (maximum of fifteen (15)). The prefix, command, and all parameters are separated by one ASCII space character (0x20) each.

The presence of a prefix is indicated with a single leading ASCII colon character (':', 0x3b), which MUST be the first character of the message itself. There MUST be NO gap (whitespace) between the colon and the prefix. The prefix is used by servers to indicate the true origin of the message. If the prefix is missing from the message, it is assumed to have originated from the connection from which it was received. Clients SHOULD not use a prefix when sending a message from themselves; if they use one, the only valid prefix is the registered nickname associated with the client.

When a server receives a message, it MUST identify its source using the (eventually assumed) prefix. If the prefix cannot be found in the server's internal database, it MUST be discarded, and if the prefix indicates the message comes from an (unknown) server, the link from which the message was received MUST be dropped. Dropping a link in such circumstances is a little excessive but necessary to maintain the integrity of the network and to prevent future problems. Another common error condition is that the prefix found in the server's internal database identifies a different source (typically a source registered from a different link than from which the message arrived). If the message was received from a server link and the prefix identifies a client, a KILL message MUST be issued for the client and sent to all servers. In other cases, the link from which the message arrived SHOULD be dropped for clients, and MUST be dropped for servers. In all cases, the message MUST be discarded.

The command MUST either be a valid IRC command or a three (3) digit number represented in ASCII text.

IRC messages are always lines of characters terminated with a CR-LF (Carriage Return - Line Feed) pair, and these messages SHALL NOT exceed 512 characters in length, counting all characters including the trailing CR-LF. Thus, there are 510 characters maximum allowed for the command and its parameters. There is no provision for continuation message lines. See section 5 for more details about current implementations.

3.3 メッセージ

サーバーとクライアントは、返信を生成する場合と生成しない場合がある場合にお互いのメッセージを送信します。サーバーは主にクライアントのルーティングタスクを実行するため、サーバー間のほとんどの通信は返信を生成しません。

各IRCメッセージは、プレフィックス（オプション）、コマンド、およびコマンドパラメーター（最大15（15））の最大3つの主要な部分で構成されている場合があります。接頭辞、コマンド、およびすべてのパラメーターは、それぞれ1つのASCIIスペース文字（0x20）で区切られています。

接頭辞の存在は、単一の主要なASCIIコロン文字（':'、0x3b）で示されています。これは、メッセージ自体の最初の文字でなければなりません。コロンと接頭辞の間にギャップ（空白）はありません。プレフィックスは、メッセージの真の起源を示すためにサーバーによって使用されます。プレフィックスがメッセージから欠落している場合、受信した接続から発信されたと想定されます。クライアントは、自分からメッセージを送信するときにプレフィックスを使用しないでください。それらを使用する場合、唯一の有効なプレフィックスは、クライアントに関連付けられた登録されたニックネームです。

サーバーがメッセージを受信した場合、（最終的に想定された）プレフィックスを使用してソースを識別する必要があります。接頭辞をサーバーの内部データベースに見つけることができない場合、破棄する必要があり、プレフィックスがメッセージを（未知の）サーバーから示す場合、メッセージが受信されたリンクをドロップする必要があります。このような状況でリンクを落とすことは少し過剰ですが、ネットワークの完全性を維持し、将来の問題を防ぐために必要です。別の一般的なエラー条件は、サーバーの内部データベースにあるプレフィックスが異なるソース（通常、メッセージが届いたものとは異なるリンクから登録されたソース）を識別することです。メッセージがサーバーリンクから受信され、プレフィックスがクライアントを識別した場合、クライアントに対してキルメッセージを発行し、すべてのサーバーに送信する必要があります。それ以外の場合、メッセージが届くリンクをクライアントのために削除する必要があり、サーバーの場合はドロップする必要があります。すべての場合において、メッセージを破棄する必要があります。

コマンドは、有効なIRCコマンドまたはASCIIテキストで表される3桁の数字でなければなりません。

IRCメッセージは、常にCR-LF（キャリッジリターン - ラインフィード）ペアで終了した文字の行の行であり、これらのメッセージは、Trailing CR-LFを含むすべての文字をカウントする長さ512文字を超えてはなりません。したがって、コマンドとそのパラメーターに許可される最大510文字があります。継続メッセージ行の規定はありません。現在の実装の詳細については、セクション5を参照してください。

3.3.1 Message format in Augmented BNF

The protocol messages must be extracted from the contiguous

3.3.1 拡張BNFのメッセージ形式

プロトコルメッセージは、オクテットの連続的なストリームか

stream of octets. The current solution is to designate two characters, CR and LF, as message separators. Empty messages are silently ignored, which permits use of the sequence CR-LF between messages without extra problems.

The extracted message is parsed into the components <prefix>, <command> and list of parameters (<params>).

The Augmented BNF representation for this is found in "IRC Client Protocol" [IRC-CLIENT].

The extended prefix (["!" user "@" host]) MUST NOT be used in server to server communications and is only intended for server to client messages in order to provide clients with more useful information about who a message is from without the need for additional queries.

3.4 Numeric replies

Most of the messages sent to the server generate a reply of some sort. The most common reply is the numeric reply, used for both errors and normal replies. The numeric reply MUST be sent as one message consisting of the sender prefix, the three digit numeric, and the target of the reply. A numeric reply is not allowed to originate from a client; any such messages received by a server are silently dropped. In all other respects, a numeric reply is just like a normal message, except that the keyword is made up of 3 numeric digits rather than a string of letters. A list of different replies is supplied in "IRC Client Protocol" [IRC-CLIENT].

4. Message Details

All the messages recognized by the IRC server and client are described in the IRC Client Protocol specification.

Where the reply ERR_NOSUCHSERVER is returned, it means that the target of the message could not be found. The server MUST NOT send any other replies after this error for that command.

The server to which a client is connected is required to parse the complete message, returning any appropriate errors. If the server encounters a fatal error while parsing a message, an error MUST be sent back to the client and the parsing terminated. A fatal error may follow from incorrect command, a destination which is otherwise unknown to the server (server, client or channel names fit this category), not enough parameters or incorrect privileges.

If a full set of parameters is presented, then each MUST be checked for validity and appropriate responses sent back to the client. In the case of messages which use parameter lists using the comma as an item separator, a reply MUST be sent for each item.

In the examples below, some messages appear using the full format:

:Name COMMAND parameter list

Such examples represent a message from "Name" in transit between servers, where it is essential to include the name of the original sender of the message so remote servers may send back a reply along the correct path.

ら抽出する必要があります。現在の解決策は、メッセージセパレーターとして2つの文字、CRとLFを指定することです。空のメッセージは静かに無視されているため、追加の問題なくメッセージ間でシーケンスCR-LFを使用できます。

抽出されたメッセージは、コンポーネント<プレフィックス>、<コマンド>、およびパラメーターのリスト（<params>）に解析されます。

このための拡張BNF表現は、「IRCクライアントプロトコル」[IRC-Client]にあります。

拡張プレフィックス（["！" user "@" host]）は、サーバー通信にサーバーに使用されてはならず、クライアントからクライアントへのメッセージのみを目的としているため、クライアントにメッセージが誰であるかについてのより有用な情報をクライアントに提供するためには、追加のクエリが必要です。

3.4 数値応答

サーバーに送信されたメッセージのほとんどは、ある種の返信を生成します。最も一般的な応答は、エラーと通常の応答の両方に使用される数値応答です。数値応答は、送信者プレフィックス、3桁の数値、および応答のターゲットで構成される1つのメッセージとして送信する必要があります。数値応答は、クライアントから発生することは許可されていません。サーバーが受信したそのようなメッセージは静かに削除されます。他のすべての点で、数値応答は通常のメッセージのようなものですが、キーワードは文字列ではなく3桁で構成されています。異なる返信のリストは、「IRCクライアントプロトコル」[IRC-Client]で提供されています。

4. メッセージの詳細

IRCサーバーとクライアントによって認識されたすべてのメッセージは、IRCクライアントプロトコル仕様で説明されています。

返信がerr_nosuchserverが返される場合、メッセージのターゲットが見つからなかったことを意味します。このコマンドのこのエラーの後、サーバーは他の返信を送信してはなりません。

クライアントが接続されているサーバーは、完全なメッセージを解析し、適切なエラーを返す必要があります。サーバーがメッセージを解析しながら致命的なエラーに遭遇した場合、エラーをクライアントに返送し、解析する解析が終了する必要があります。誤ったエラーは、サーバーに不明な宛先（このカテゴリに適合するサーバー、クライアント、またはチャネル名が適合）から、誤ったエラーが続く場合があります。

パラメーターの完全なセットが提示されている場合、それぞれに妥当性とクライアントに返信される適切な応答があることを確認する必要があります。コンマをアイテムセパレーターとして使用してパラメーターリストを使用するメッセージの場合、各アイテムに対して返信を送信する必要があります。

以下の例では、いくつかのメッセージが完全な形式を使用して表示されます。

：名前コマンドパラメーターリスト

このような例は、サーバー間のトランジットの「名前」からのメッセージを表しています。ここでは、リモートサーバーが正しいパスに沿って返信を送信できるように、メッセージの元の送信者の名前を含めることが不可欠です。

The message details for client to server communication are described in the "IRC Client Protocol" [IRC-CLIENT]. Some sections in the following pages apply to some of these messages, they are additions to the message specifications which are only relevant to server to

server communication, or to the server implementation. The messages which are introduced here are only used for server to server communication.

クライアントからサーバーへの通信のメッセージの詳細は、「IRCクライアントプロトコル」[IRC-Client]で説明されています。次のページの一部のセクションは、これらのメッセージの一部に適用されます。これらは、サーバーにのみ関連するメッセージ仕様に追加されています。

サーバー通信、またはサーバーの実装。ここで導入されたメッセージは、サーバーからサーバー通信にのみ使用されます。

4.1 Connection Registration

The commands described here are used to register a connection with another IRC server.

4.1 接続登録

ここで説明するコマンドは、別のIRCサーバーとの接続を登録するために使用されます。

4.1.1 Password message

4.1.1 パスワードメッセージ

```
    Command: PASS
 Parameters: <password> <version> <flags> [<options>]
```

The PASS command is used to set a 'connection password'. The password MUST be set before any attempt to register the connection is made. Currently this means that servers MUST send a PASS command before any SERVER command. Only one (1) PASS command SHALL be accepted from a connection.

パスコマンドは、「接続パスワード」を設定するために使用されます。接続を登録する試みが行われる前に、パスワードを設定する必要があります。現在、これは、サーバーがサーバーコマンドの前にパスコマンドを送信する必要があることを意味します。接続から受け入れられるものとする1つのパスコマンドのみが受け入れられます。

The last three (3) parameters MUST be ignored if received from a client (e.g. a user or a service). They are only relevant when received from a server.

クライアント（ユーザーまたはサービスなど）から受信した場合、最後の3つのパラメーターを無視する必要があります。サーバーから受信した場合にのみ関連します。

The <version> parameter is a string of at least four (4) characters, and up to fourteen (14) characters. The first four (4) characters MUST be digits and indicate the protocol version known by the server issuing the message. The protocol described by this document is version 2.10 which is encoded as "0210". The remaining OPTIONAL characters are implementation dependent and should describe the software version number.

<バージョン>パラメーターは、少なくとも4文字の文字列であり、最大14文字です。最初の4文字は数字であり、メッセージを発行するサーバーが既知のプロトコルバージョンを示す必要があります。このドキュメントで説明されているプロトコルは、「0210」としてエンコードされているバージョン2.10です。残りのオプション文字は実装依存であり、ソフトウェアバージョン番号を説明する必要があります。

The <flags> parameter is a string of up to one hundred (100) characters. It is composed of two substrings separated by the character "|" (%x7C). If present, the first substring MUST be the name of the implementation. The reference implementation (See Section 8, "Current support and availability") uses the string "IRC". If a different implementation is written, which needs an identifier, then that identifier should be registered through publication of an RFC. The second substring is implementation dependent. Both substrings are OPTIONAL, but the character "|" is REQUIRED. The character "|" MUST NOT appear in either substring.

<Flags>パラメーターは、最大100文字の文字列です。文字「|」で区切られた2つのサブストリングで構成されています（%x7c）。存在する場合、最初のサブストリングは実装の名前でなければなりません。参照実装（セクション8、「現在のサポートと可用性」を参照）は、文字列「IRC」を使用します。識別子が必要な別の実装が書かれている場合、その識別子はRFCの公開を通じて登録する必要があります。2番目のサブストリングは実装依存です。両方のサブストリングはオプションですが、文字「|」必要とされている。キャラクター「|」どちらのサブストリングにも表示されないでください。

Finally, the last parameter, <options>, is used for link options. The only options defined by the protocol are link compression (using the character "Z"), and an abuse protection flag (using the character

"P"). See sections 5.3.1.1 (Compressed server to server links) and 5.3.1.2 (Anti abuse protections) respectively for more information on these options.

最後に、最後のパラメーター<オプション>は、リンクオプションに使用されます。プロトコルで定義される唯一のオプションは、リンク圧縮（文字「Z」を使用）と乱用保護フラグ（文字を使用している）です。

「P」）。これらのオプションの詳細については、それぞれセクション5.3.1.1（サーバーからサーバーリンクから圧縮サーバーへのリンク）と5.3.1.2（防止防止保護）を参照してください。

Numeric Replies:

ERR_NEEDMOREPARAMS ERR_ALREADYREGISTRED

数値応答：

err_needmoreparams err_alreadyregistred

Example:

例：

PASS moresecretpassword 0210010000 IRC|aBgH$ Z

Pass moresecretpassword 0210010000 IRC | ABGH $ Z

4.1.2 Server message

4.1.2 サーバーメッセージ

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

The SERVER command is used to register a new server. A new connection introduces itself as a server to its peer. This message is also used to pass server data over whole net. When a new server is connected to net, information about it MUST be broadcasted to the whole network.

サーバーコマンドは、新しいサーバーの登録に使用されます。新しい接続は、サーバーとしてピアに自己導入されます。このメッセージは、ネット全体にサーバーデータを渡すためにも使用されます。新しいサーバーがネットに接続されている場合、それに関する情報はネットワーク全体にブロードキャストする必要があります。

The <info> parameter may contain space characters.

<info>パラメーターにはスペース文字が含まれている場合があります。

<hopcount> is used to give all servers some internal information on how far away each server is. Local peers have a value of 0, and each passed server increments the value. With a full server list, it would be possible to construct a map of the entire server tree, but hostmasks prevent this from being done.

<HopCount>は、すべてのサーバーに各サーバーがどれだけ離れているかについての内部情報を提供するために使用されます。ローカルピアの値は0で、各渡されたサーバーは値を増やします。完全なサーバーリストを使用すると、サーバーツリー全体のマップを作成することができますが、ホストマスクはこれが行われないようにします。

The <token> parameter is an unsigned number used by servers as an identifier. This identifier is subsequently used to reference a server in the NICK and SERVICE messages sent between servers. Server tokens only have a meaning for the point-to-point peering they are used and MUST be unique for that connection. They are not global.

<token>パラメーターは、サーバーが識別子として使用する署名のない番号です。その後、この識別子は、ニック内のサーバーとサーバー間で送信されるサービスメッセージのサーバーを参照するために使用されます。サーバートークンには、使用されるポイントツーポイントピアリングの意味があり、その接続には一意でなければなりません。彼らはグローバルではありません。

The SERVER message MUST only be accepted from either (a) a connection which is yet to be registered and is attempting to register as a server, or (b) an existing connection to another server, in which case the SERVER message is introducing a new server behind that server.

サーバーメッセージは、（a）まだ登録されておらず、サーバーとして登録しようとしている接続、または（b）別のサーバーへの既存の接続を受け入れている場合のみを受け入れる必要があります。その場合、サーバーメッセージは新しいものを導入していますそのサーバーの背後にあるサーバー。

Most errors that occur with the receipt of a SERVER command result in the connection being terminated by the destination host (target SERVER). Because of the severity of such event, error replies are usually sent using the "ERROR" command rather than a numeric.

サーバーコマンドの受領で発生するほとんどのエラーは、宛先ホスト（ターゲットサーバー）によって接続が終了するようになります。このようなイベントの重大度のため、通常、数値ではなく「エラー」コマンドを使用してエラー応答が送信されます。

If a SERVER message is parsed and it attempts to introduce a server which is already known to the receiving server, the connection, from which that message arrived, MUST be closed (following the correct procedures), since a duplicate route to a server has been formed and the acyclic nature of the IRC tree breaks. In some conditions, the connection from which the already known server has registered MAY be closed instead. It should be noted that this kind of error can also be the result of a second running server, problem which cannot be fixed within the protocol and typically requires human intervention. This type of problem is particularly insidious, as it can quite easily result in part of the IRC network to be isolated, with one of the two servers connected to each partition therefore making it impossible for the two parts to unite.

サーバーメッセージが解析され、受信サーバーに既に知られているサーバーを導入しようとする場合、そのメッセージが届く接続が閉じている必要があります（正しい手順に従ってください）サーバーへのルートが重複しているためです形成され、IRCツリーの非環式性。一部の条件では、既知のサーバーが登録している接続が代わりに閉じられる場合があります。この種のエラーは、プロトコル内で修正できず、通常は人間の介入を必要とする2番目の実行サーバーの結果である可能性があることに注意する必要があります。このタイプの問題は特に陰湿です。IRCネットワークの一部を分離する可能性が非常に高いため、2つのサーバーのいずれかが各パーティションに接続されているため、2つの部分が団結することは不可能です。

Numeric Replies:

数値応答：

ERR_ALREADYREGISTRED

ERR_ALREADYREGISTRED

Example:

例：

SERVER test.oulu.fi 1 1 :Experimental server ; New server test.oulu.fi introducing itself and attempting to register.

Server Test.oulu.fi 1 1：実験サーバー。New Server Test.oulu.fiは、それ自体を紹介し、登録しようとしています。

:tolsun.oulu.fi SERVER csd.bu.edu 5 34 :BU Central Server ; Server tolsun.oulu.fi is our uplink for csd.bu.edu which is 5 hops away. The token "34" will be used by tolsun.oulu.fi when introducing new users or services connected to csd.bu.edu.

：tolsun.oulu.fiサーバーcsd.bu.edu 5 34：bu Central Server;Server Tolsun.oulu.fiは、CSD.bu.eduのアップリンクです。トークン「34」は、csd.bu.eduに接続されている新しいユーザーまたはサービスを紹介するときにtolsun.oulu.fiによって使用されます。

4.1.3 Nick

4.1.3 ニック

```
     Command: NICK
  Parameters: <nickname> <hopcount> <username> <host> <servertoken>
              <umode> <realname>
```

This form of the NICK message MUST NOT be allowed from user connections. However, it MUST be used instead of the NICK/USER pair to notify other servers of new users joining the IRC network.

この形式のニックメッセージは、ユーザー接続から許可されてはなりません。ただし、IRCネットワークに参加する新しいユーザーの他のサーバーに通知するには、Nick/ユーザーペアの代わりに使用する必要があります。

This message is really the combination of three distinct messages: NICK, USER and MODE [IRC-CLIENT].

このメッセージは、実際には、ニック、ユーザー、モード[IRC-Client]の3つの異なるメッセージの組み合わせです。

The <hopcount> parameter is used by servers to indicate how far away a user is from its home server. A local connection has a hopcount of 0. The hopcount value is incremented by each passed server.

<HopCount>パラメーターは、ユーザーがホームサーバーからどれだけ離れているかを示すためにサーバーによって使用されます。ローカル接続のホップカウントは0です。ホップカウント値は、通過する各サーバーによって増加します。

The <servertoken> parameter replaces the <servername> parameter of the USER (See section 4.1.2 for more information on server tokens).

<Servertoken>パラメーターは、ユーザーの<Servername>パラメーターを置き換えます（サーバートークンの詳細については、セクション4.1.2を参照）。

Examples:

例：

NICK syrk 5 kalt millennium.stealth.net 34 +i :Christophe Kalt ; New user with nickname "syrk", username "kalt", connected from host "millennium.stealth.net" to server "34" ("csd.bu.edu" according to the previous example).

Nick Syrk 5 Kalt Millennium.stealth.net 34 I：クリストフカルト; ニックネーム「Syrk」、ユーザー名「Kalt」を持つ新しいユーザー、ホスト「Millennium.stealth.net」からサーバー「34」に接続されています（前の例に従って「csd.bu.edu」）。

:krys NICK syrk ; The other form of the NICK message, as defined in "IRC Client Protocol" [IRC-CLIENT] and used between servers: krys changed his nickname to syrk

：Krys Nick Syrk;「IRCクライアントプロトコル」[IRC-Client]で定義され、サーバー間で使用されるニックメッセージの他の形式：Krysはニックネームをシルクに変更しました

4.1.4 Service message

4.1.4 サービスメッセージ

```
     Command: SERVICE
  Parameters: <servicename> <servertoken> <distribution> <type>
               <hopcount> <info>
```

The SERVICE command is used to introduce a new service. This form of the SERVICE message SHOULD NOT be allowed from client (unregistered, or registered) connections. However, it MUST be used between servers to notify other servers of new services joining the IRC network.

サービスコマンドは、新しいサービスを導入するために使用されます。この形式のサービスメッセージは、クライアント（未登録、または登録済み）接続から許可されないでください。ただし、サーバー間で使用して、IRCネットワークに参加する新しいサービスの他のサーバーに通知する必要があります。

The <servertoken> is used to identify the server to which the service is connected. (See section 4.1.2 for more information on server tokens).

<servertoken>は、サービスが接続されているサーバーを識別するために使用されます。（サーバートークンの詳細については、セクション4.1.2を参照してください）。

The <hopcount> parameter is used by servers to indicate how far away a service is from its home server. A local connection has a hopcount of 0. The hopcount value is incremented by each passed server.

<HopCount>パラメーターは、サービスがホームサーバーからどれだけ離れているかを示すためにサーバーによって使用されます。ローカル接続のホップカウントは0です。ホップカウント値は、通過する各サーバーによって増加します。

The <distribution> parameter is used to specify the visibility of a service. The service may only be known to servers which have a name matching the distribution. For a matching server to have knowledge of the service, the network path between that

<distribution>パラメーターは、サービスの可視性を指定するために使用されます。このサービスは、分布に一致する名前のサーバーに対してのみ知られている場合があります。一致するサーバーがサービスの知識を持つために、そのサーバーとサービ

server and the server to which the service is connected MUST be composed of servers whose names all match the mask. Plain "*" is used when no restriction is wished.

The <type> parameter is currently reserved for future usage.

Numeric Replies:

ERR_ALREADYREGISTRED ERR_NEEDMOREPARAMS ERR_ERRONEUSNICKNAME RPL_YOURESERVICE RPL_YOURHOST RPL_MYINFO

Example:

SERVICE dict@irc.fr 9 *.fr 0 1 :French Dictionary r" registered on server "9" is being announced to another server. This service will only be available on servers whose name matches "*.fr".

スが接続されているサーバーとの間のネットワークパスは、名前がすべてマスクと一致するサーバーで構成する必要があります。単純な「*」は、制限が希望されない場合に使用されます。

<type>パラメーターは現在、将来の使用のために予約されています。

数値応答：

err_alreadyregistred err_needmoreparams err_erroneusnickname rpl_youreservice rpl_yourhost rpl_myinfo

例：

Service dict@irc.fr 9 *.fr 0 1：フランス語辞書r "サーバーに登録されている" 9 "は別のサーバーに発表されます。このサービスは、名前が「 *.fr」と一致するサーバーでのみ利用できます。

4.1.5 Quit

4.1.5 終了する

```
     Command: QUIT
  Parameters: [<Quit Message>]
```

A client session ends with a quit message. The server MUST close the connection to a client which sends a QUIT message. If a "Quit Message" is given, this will be sent instead of the default message, the nickname or service name.

When "netsplit" (See Section 4.1.6) occur, the "Quit Message" is composed of the names of two servers involved, separated by a space. The first name is that of the server which is still connected and the second name is either that of the server which has become disconnected or that of the server to which the leaving client was connected:

クライアントセッションは、終了メッセージで終了します。サーバーは、終了メッセージを送信するクライアントへの接続を閉じる必要があります。「Quitメッセージ」が与えられた場合、これはデフォルトのメッセージ、ニックネーム、またはサービス名の代わりに送信されます。

「netsplit」（セクション4.1.6を参照）が発生すると、「quitメッセージ」は、スペースで区切られた2つのサーバーの名前で構成されます。最初の名前は、まだ接続されているサーバーの名前であり、2番目の名前は、切断されたサーバーの名前または退職クライアントが接続されているサーバーのものです。

```
      <Quit Message> =  ":" servername SPACE servername
```

Because the "Quit Message" has a special meaning for "netsplits", servers SHOULD NOT allow a client to use a <Quit Message> in the format described above.

If, for some other reason, a client connection is closed without the client issuing a QUIT command (e.g. client dies and EOF occurs on socket), the server is REQUIRED to fill in the quit message with some sort of message reflecting the nature of the event which caused it to happen. Typically, this is done by reporting a system specific error.

「Quitメッセージ」には「Netsplits」に対して特別な意味があるため、サーバーは上記の形式でクライアントが<Quitメッセージ>を使用できるようにしてはなりません。

他の理由で、クライアントがQuitコマンドを発行せずにクライアント接続が閉じられている場合（たとえば、クライアントが死亡し、EOFがソケットで発生します）、サーバーは、QUITメッセージに何らかのメッセージを入力する必要があります。それを引き起こしたイベント。通常、これはシステム固有のエラーを報告することによって行われます。

Numeric Replies:

None.

Examples:

:WiZ QUIT :Gone to have lunch ; Preferred message format.

数値応答：

なし。

例：

：wiz quit：昼食をとった。優先メッセージ形式。

4.1.6 Server quit message

4.1.6 サーバーはメッセージを終了します

```
     Command: SQUIT
  Parameters: <server> <comment>
```

The SQUIT message has two distinct uses.

The first one (described in "Internet Relay Chat: Client Protocol" [IRC-CLIENT]) allows operators to break a local or remote server link. This form of the message is also eventually

Squitメッセージには2つの異なる用途があります。

最初のもの（「インターネットリレーチャット：クライアントプロトコル」[IRC-Client]で説明されています）を使用すると、オペレーターはローカルまたはリモートサーバーのリンクを破

used by servers to break a remote server link.

ることができます。この形式のメッセージは、最終的にサーバーによってリモートサーバーリンクを破るために使用されます。

The second use of this message is needed to inform other servers when a "network split" (also known as "netsplit") occurs, in other words to inform other servers about quitting or dead servers. If a server wishes to break the connection to another server it MUST send a SQUIT message to the other server, using the name of the other server as the server parameter, which then closes its connection to the quitting server.

このメッセージの2番目の使用は、「ネットワークスプリット」（「netsplit」とも呼ばれる）が発生したときに他のサーバーを通知するために、つまり他のサーバーに終了または死んだサーバーについて通知するために必要です。サーバーが別のサーバーへの接続を壊したい場合は、他のサーバーの名前をサーバーパラメーターとして使用して、他のサーバーにスクイットメッセージを送信する必要があります。

The <comment> is filled in by servers which SHOULD place an error or similar message here.

<コメント>は、ここにエラーまたは同様のメッセージを配置するサーバーによって記入されています。

Both of the servers which are on either side of the connection being closed are REQUIRED to send out a SQUIT message (to all its other server connections) for all other servers which are considered to be behind that link.

接続の両側にあるサーバーの両方は、そのリンクの背後にあると考えられている他のすべてのサーバーのスクイットメッセージ（他のすべてのサーバー接続）を送信する必要があります。

Similarly, a QUIT message MAY be sent to the other still connected servers on behalf of all clients behind that quitting link. In addition to this, all channel members of a channel which lost a member due to the "split" MUST be sent a QUIT message. Messages to channel members are generated by each client's local server.

同様に、QUITメッセージは、その終了リンクの背後にあるすべてのクライアントに代わって、他のまだ接続されたサーバーに送信される場合があります。これに加えて、「分割」のためにメンバーを失ったチャネルのすべてのチャネルメンバーに終了メッセージが送信される必要があります。チャネルメンバーへのメッセージは、各クライアントのローカルサーバーによって生成されます。

If a server connection is terminated prematurely (e.g., the server on the other end of the link died), the server which detects this disconnection is REQUIRED to inform the rest of the network that the connection has closed and fill in the comment field with something appropriate.

サーバー接続が時期尚早に終了した場合（たとえば、リンクの反対側のサーバーが死んだ）、この切断を検出するサーバーは、ネットワークの残りの部分に接続が閉じていることを通知し、コメントフィールドに何かを入力します適切な。

When a client is removed as the result of a SQUIT message, the server SHOULD add the nickname to the list of temporarily unavailable nicknames in an attempt to prevent future nickname collisions. See

スクイットメッセージの結果としてクライアントが削除されると、サーバーは、将来のニックネームの衝突を防ぐために、一時的に利用できないニックネームのリストにニックネームを追加する必要があります。見る

section 5.7 (Tracking recently used nicknames) for more information on this procedure.

この手順の詳細については、セクション5.7（最近使用されたニックネームの追跡）。

Numeric replies:

数値応答：

ERR_NOPRIVILEGES ERR_NOSUCHSERVER ERR_NEEDMOREPARAMS

err_noprivileges err_nosuchserver err_needmoreparams

Example:

例：

SQUIT tolsun.oulu.fi :Bad Link ? ; the server link tolson.oulu.fi has been terminated because of "Bad Link".

Squit Tolsun.oulu.fi：リンクが悪いですか？;サーバーリンクtolson.oulu.fiは、「悪いリンク」のために終了しました。

:Trillian SQUIT cm22.eng.umd.edu :Server out of control ; message from Trillian to disconnect "cm22.eng.umd.edu" from the net because "Server out of control".

：Trillian Squit CM22.eng.umd.edu：server of of Control;トリリアンからのメッセージは、「cm22.eng.umd.edu」をネットから切断します。これは、「サーバーが制御できない」ためです。

4.2 Channel operations

4.2 チャネル操作

This group of messages is concerned with manipulating channels, their properties (channel modes), and their contents (typically users). In implementing these, a number of race conditions are inevitable when users at opposing ends of a network send commands which will ultimately clash. It is also REQUIRED that servers keep a nickname history to ensure that wherever a <nick> parameter is given, the server check its history in case it has recently been changed.

このメッセージのグループは、チャネルの操作、そのプロパティ（チャネルモード）、およびその内容（通常はユーザー）に関係しています。これらを実装するには、最終的に衝突するネットワークの対立する端部のユーザーがコマンドを送信する場合、多くのレース条件が避けられません。また、サーバーがニックネームの履歴を保持して、<Nick>パラメーターが提供されている場合は、最近変更された場合にサーバーがその履歴を確認することも必要です。

4.2.1 Join message

4.2.1 メッセージに参加します

```
     Command: JOIN
  Parameters: <channel>[ %x7 <modes> ]
              *( "," <channel>[ %x7 <modes> ] )
```

The JOIN command is used by client to start listening a specific channel. Whether or not a client is allowed to join a channel is checked only by the local server the client is connected to; all other servers automatically add the user to the channel when the command is received from other servers.

Joinコマンドは、特定のチャネルのリスニングを開始するためにクライアントが使用します。クライアントがチャネルに参加できるかどうかは、クライアントが接続されているローカルサーバーによってのみチェックされます。他のすべてのサーバーは、他のサーバーからコマンドが受信されたときに、自動的にユーザーをチャネルに追加します。

Optionally, the user status (channel modes 'O', 'o', and 'v') on the channel may be appended to the channel name using a control G (^G or ASCII 7) as separator. Such data MUST be ignored if the message wasn't received from a server. This format MUST NOT be sent to clients, it can only be used between servers and SHOULD be avoided.

オプションで、チャネル上のユーザーステータス（チャネルモード 'o'、 'o'、および 'v'）は、セパレーターとしてコントロールg（^gまたはascii 7）を使用してチャネル名に追加できます。メッセージがサーバーから受信されなかった場合、そのようなデータは無視する必要があります。この形式はクライアントに送信してはなりません。サーバー間でのみ使用でき、避ける必要があります。

The JOIN command MUST be broadcast to all servers so that each server knows where to find the users who are on the channel. This allows optimal delivery of PRIVMSG and NOTICE messages to the channel.

Joinコマンドは、各サーバーがチャネル上のユーザーを見つける場所を把握できるように、すべてのサーバーにブロードキャストする必要があります。これにより、Privmsgの最適な配信とチャネルへのメッセージが表示されます。

Numeric Replies:

数値応答：

ERR_NEEDMOREPARAMS ERR_BANNEDFROMCHAN
ERR_INVITEONLYCHAN ERR_BADCHANNELKEY
ERR_CHANNELISFULL ERR_BADCHANMASK
ERR_NOSUCHCHANNEL ERR_TOOMANYCHANNELS
ERR_TOOMANYTARGETS ERR_UNAVAILRESOURCE
RPL_TOPIC

err_needmoreparams err_bannedfromchan
err_inviteonlychan err_badchannelkey err_channelisfull
err_badchanmask err_nosuchchannel err_toomanychannels
err_toomanytargets err_unavailresource rpl_topicicic

Examples:

例：

```
  :WiZ JOIN #Twilight_zone        ; JOIN message from WiZ
```

4.2.2 Njoin message

4.2.2 nジョインメッセージ

```
     Command: NJOIN
  Parameters: <channel> [ "@@" / "@" ] [ "+" ] <nickname>
                        *( "," [ "@@" / "@" ] [ "+" ] <nickname> )
```

The NJOIN message is used between servers only. If such a message is received from a client, it MUST be ignored. It is used when two servers connect to each other to exchange the list of channel members for each channel.

NJOINメッセージは、サーバー間でのみ使用されます。そのようなメッセージがクライアントから受信された場合、それは無視する必要があります。2つのサーバーが相互に接続して、各チャネルのチャネルメンバーのリストを交換するときに使用されます。

Even though the same function can be performed by using a succession of JOIN, this message SHOULD be used instead as it is more efficient. The prefix "@@" indicates that the user is the "channel creator", the character "@" alone indicates a "channel operator", and the character '+' indicates that the user has the voice privilege.

一連の結合を使用して同じ関数を実行できますが、このメッセージはより効率的であるため、代わりに使用する必要があります。接頭辞「@@」は、ユーザーが「チャネル作成者」であることを示し、キャラクター「@」のみが「チャネル演算子」を示し、文字「ユーザーが音声特権を持っていることを示します。

Numeric Replies:

数値応答：

ERR_NEEDMOREPARAMS ERR_NOSUCHCHANNEL
ERR_ALREADYREGISTRED

err_needmoreparams err_nosuchchannel
err_alreadyregistred

Examples:

例：

:ircd.stealth.net NJOIN #Twilight_zone :@WiZ,+syrk,avalon ; NJOIN message from ircd.stealth.net announcing users joining

：ircd.stealth.net njoin #twilight_zone：@wiz、syrk、avalon;IRCD.stealth.netからのnjoinメッセージ#twilight_zoneチ

the #Twilight_zone channel: WiZ with channel operator status, syrk with voice privilege and avalon with no privilege.

ャンネルに参加するユーザーの発表：チャネルオペレータースステータス、音声特権を持つシルク、特権のないアバロン。

4.2.3 Mode message

The MODE message is a dual-purpose command in IRC. It allows both usernames and channels to have their mode changed.

When parsing MODE messages, it is RECOMMENDED that the entire message be parsed first, and then the changes which resulted passed on.

It is REQUIRED that servers are able to change channel modes so that "channel creator" and "channel operators" may be created.

4.2.3 モードメッセージ

モードメッセージは、IRCの二重目的コマンドです。ユーザー名とチャネルの両方がモードを変更できるようにします。

モードメッセージを解析するときは、メッセージ全体を最初に解析し、次に結果の変更が渡されることをお勧めします。

サーバーがチャネルモードを変更して、「チャネル作成者」と「チャネルオペレーター」を作成できるようにする必要があります。

5. Implementation details

A the time of writing, the only current implementation of this protocol is the IRC server, version 2.10. Earlier versions may implement some or all of the commands described by this document with NOTICE messages replacing many of the numeric replies. Unfortunately, due to backward compatibility requirements, the implementation of some parts of this document varies with what is laid out. One notable difference is:

* recognition that any LF or CR anywhere in a message marks the end of that message (instead of requiring CR-LF);

The rest of this section deals with issues that are mostly of importance to those who wish to implement a server but some parts also apply directly to clients as well.

5. 実装の詳細

執筆の時間、このプロトコルの現在の実装のみはIRCサーバーであるバージョン2.10です。以前のバージョンは、このドキュメントで説明されているコマンドの一部またはすべてを実装して、多くの数値応答を置き換える通知メッセージを実装する場合があります。残念ながら、後方互換性の要件により、このドキュメントの一部のいくつかの部分の実装は、レイアウトされたものによって異なります。顕著な違いの1つは、次のとおりです。

* メッセージ内のどこでもLFまたはCRがそのメッセージの終わりをマークすることを認識します（CR-LFを必要とする代わりに）。

このセクションの残りの部分では、サーバーを実装したい人にとって主に重要な問題を扱っていますが、一部の部品もクライアントに直接適用します。

5.1 Connection 'Liveness'

To detect when a connection has died or become unresponsive, the server MUST poll each of its connections. The PING command (See "IRC Client Protocol" [IRC-CLIENT]) is used if the server doesn't get a response from its peer in a given amount of time.

If a connection doesn't respond in time, its connection is closed using the appropriate procedures.

5.1 接続「活性」

接続が死んだり、反応しなくなったりしたときに検出するには、サーバーがそれぞれの接続を投票する必要があります。Pingコマンド（「IRCクライアントプロトコル」[IRC-Client]を参照）は、サーバーが特定の時間でピアから応答を取得しない場合に使用されます。

接続が時間内に応答しない場合、適切な手順を使用して接続が閉じられます。

5.2 Accepting a client to server connection

5.2.1 Users

When a server successfully registers a new user connection, it is REQUIRED to send to the user unambiguous messages stating: the user identifiers upon which it was registered (RPL_WELCOME), the server name and version (RPL_YOURHOST), the server birth information (RPL_CREATED), available user and channel modes (RPL_MYINFO), and it MAY send any introductory messages which may be deemed appropriate.

In particular the server SHALL send the current user/service/server count (as per the LUSER reply) and finally the MOTD (if any, as per the MOTD reply).

After dealing with registration, the server MUST then send out to other servers the new user's nickname (NICK message), other information as supplied by itself (USER message) and as the server could discover (from DNS servers). The server

5.2 サーバー接続へのクライアントを受け入れます

5.2.1 ユーザー

サーバーが新しいユーザー接続を正常に登録する場合、ユーザーに登録されたユーザー識別子（rpl_welcome）、サーバー名とバージョン（rpl_yourhost）、サーバーの誕生情報（rpl_created）を記載するユーザーに送信する必要があります。、利用可能なユーザーおよびチャネルモード（RPL_MYINFO）、および適切とみなされる可能性のある入門メッセージを送信する場合があります。

特に、サーバーは、現在のユーザー/サービス/サーバーカウント（Luser Replyに従って）を送信し、最後にMOTD（MOTD返信に従って）を送信します。

登録を処理した後、サーバーは他のサーバーに新しいユーザーのニックネーム（Nickメッセージ）、その他の情報自体（ユーザーメッセージ）、およびサーバーが発見できるように（DNSサーバーから）送信する必要があります。サーバーは、「IRCク

MUST NOT send this information out with a pair of NICK and USER messages as defined in "IRC Client Protocol" [IRC-CLIENT], but MUST instead take advantage of the extended NICK message defined in section 4.1.3.

ライアントプロトコル」[IRC-Client]で定義されているニックとユーザーメッセージのペアでこの情報を送信してはなりませんが、代わりにセクション4.1.3で定義されている拡張ニックメッセージを利用する必要があります。

5.2.2 Services

Upon successfully registering a new service connection, the server is subject to the same kind of REQUIREMENTS as for a user. Services being somewhat different, only the following replies are sent: RPL_YOURESERVICE, RPL_YOURHOST, RPL_MYINFO.

5.2.2 サービス

新しいサービス接続を正常に登録すると、サーバーはユーザーと同じ種類の要件の対象となります。サービスは多少異なり、次の返信のみが送信されます：rpl_youreservice、rpl_yourhost、rpl_myinfo。

After dealing with this, the server MUST then send out to other servers (SERVICE message) the new service's nickname and other information as supplied by the service (SERVICE message) and as the server could discover (from DNS servers).

これを処理した後、サーバーは他のサーバー（サービスメッセージ）に新しいサービスのニックネームと、サービス（サービスメッセージ）から提供され、サーバーが発見できるように（DNSサーバーから）他の情報を送信する必要があります。

5.3 Establishing a server-server connection.

5.3 サーバーサーバー接続の確立。

The process of establishing a server-to-server connection is fraught with danger since there are many possible areas where problems can occur - the least of which are race conditions.

サーバー間接続を確立するプロセスは、問題が発生する可能性のある多くの領域があるため、危険に満ちています。

After a server has received a connection following by a PASS/SERVER pair which were recognized as being valid, the server SHOULD then reply with its own PASS/SERVER information for that connection as well as all of the other state information it knows about as described below.

サーバーが有効であると認識されたパス/サーバーペアのフォローを受信した後、サーバーはその接続の独自のパス/サーバー情報と、説明されている他のすべての状態情報で返信する必要があります下。

When the initiating server receives a PASS/SERVER pair, it too then checks that the server responding is authenticated properly before accepting the connection to be that server.

開始サーバーがパス/サーバーのペアを受信すると、そのサーバーがそのサーバーへの接続を受け入れる前に、サーバーの応答が適切に認証されていることを確認します。

5.3.1 Link options

Server links are based on a common protocol (defined by this document) but a particular link MAY set specific options using the PASS message (See Section 4.1.1).

5.3.1 リンクオプション

サーバーリンクは共通のプロトコル（このドキュメントで定義）に基づいていますが、特定のリンクはパスメッセージを使用して特定のオプションを設定する場合があります（セクション4.1.1を参照）。

5.3.1.1 Compressed server to server links

If a server wishes to establish a compressed link with its peer, it MUST set the 'Z' flag in the options parameter to the PASS message. If both servers request compression and both servers are able to initialize the two compressed streams, then the remainder of the communication is to be compressed. If any server fails to initialize the stream, it will send an uncompressed ERROR message to its peer and close the connection.

5.3.1.1 サーバーからサーバーリンクへの圧縮

サーバーがピアで圧縮リンクを確立したい場合、オプションパラメーターの「z」フラグをパスメッセージに設定する必要があります。両方のサーバーが圧縮を要求し、両方のサーバーが2つの圧縮ストリームを初期化できる場合、残りの通信を圧縮します。サーバーがストリームの初期化に失敗した場合、非圧縮エラーメッセージがピアに送信され、接続を閉じます。

The data format used for the compression is described by RFC 1950 [ZLIB], RFC 1951 [DEFLATE] and RFC 1952 [GZIP].

圧縮に使用されるデータ形式は、RFC 1950 [ZLIB]、RFC 1951 [DEFLATE]、RFC 1952 [GZIP]によって説明されています。

5.3.1.2 Anti abuse protections

Most servers implement various kinds of protections against possible abusive behaviours from non trusted parties (typically users). On some networks, such protections are indispensable, on others they are superfluous. To require that all servers implement and enable such features on a particular network, the 'P' flag is used when two servers connect. If this flag is present, it means that the server protections are enabled, and that the server REQUIRES all its server links to enable them as well.

5.3.1.2 虐待防止保護

ほとんどのサーバーは、信頼できない当事者（通常はユーザー）からの虐待的な行動に対するさまざまな種類の保護を実装しています。一部のネットワークでは、そのような保護は不可欠であり、他のネットワークでは不要です。すべてのサーバーが特定のネットワーク上にそのような機能を実装および有効にすることを要求するために、2つのサーバーが接続すると「P」フラグが使用されます。このフラグが存在する場合、それはサーバー保護が有効になっていることを意味し、サーバーがすべてのサーバーリンクを必要としてそれらを有効にすることも意味します。

Commonly found protections are described in sections 5.7

一般的に見つかった保護は、セクション5.7（最近使用されたニ

(Tracking recently used nicknames) and 5.8 (Flood control of clients).

5.3.2 State information exchange when connecting

The order of state information being exchanged between servers is essential. The REQUIRED order is as follows:

* all known servers;

* all known client information;

* all known channel information.

Information regarding servers is sent via extra SERVER messages, client information with NICK and SERVICE messages and channels with NJOIN/MODE messages.

NOTE: channel topics SHOULD NOT be exchanged here because the TOPIC command overwrites any old topic information, so at best, the two sides of the connection would exchange topics.

By passing the state information about servers first, any collisions with servers that already exist occur before nickname collisions caused by a second server introducing a particular nickname. Due to the IRC network only being able to exist as an acyclic graph, it may be possible that the network has already reconnected in another location. In this event, the place where the server collision occurs indicates where the net needs to split.

5.4 Terminating server-client connections

When a client connection unexpectedly closes, a QUIT message is generated on behalf of the client by the server to which the client was connected. No other message is to be generated or used.

5.5 Terminating server-server connections

If a server-server connection is closed, either via a SQUIT command or "natural" causes, the rest of the connected IRC network MUST have its information updated by the server which detected the closure. The terminating server then sends a list of SQUITs (one for each server behind that connection). (See Section 4.1.6 (SQUIT)).

5.6 Tracking nickname changes

All IRC servers are REQUIRED to keep a history of recent nickname changes. This is important to allow the server to have a chance of keeping in touch of things when nick-change race conditions occur with commands manipulating them. Messages which MUST trace nick changes are:

* KILL (the nick being disconnected)

* MODE (+/- o,v on channels)

* KICK (the nick being removed from channel)

No other commands need to check nick changes.

In the above cases, the server is required to first check for the existence of the nickname, then check its history to see who that nick now belongs to (if anyone!). This reduces the chances of race conditions but they can still occur with the server

ックネームの追跡）と5.8（クライアントの洪水制御）で説明されています。

5.3.2 接続時の状態情報交換

サーバー間で交換される州の情報の順序は不可欠です。必要な注文は次のとおりです。

* すべての既知のサーバー。

* すべての既知のクライアント情報。

* すべての既知のチャネル情報。

サーバーに関する情報は、追加のサーバーメッセージ、ニックを備えたクライアント情報、およびNJOIN/モードメッセージを使用したサービス、およびチャネルを介して送信されます。

注：トピックコマンドが古いトピック情報を上書きしているため、チャネルトピックをここで交換しないでください。せいぜい、接続の2つの側面がトピックを交換します。

最初にサーバーに関する状態情報を渡すことにより、すでに存在するサーバーとの衝突は、特定のニックネームを導入する2番目のサーバーによって引き起こされる衝突の前に発生します。IRCネットワークは非環式グラフとしてのみ存在できるため、ネットワークがすでに別の場所で再接続されている可能性があります。この場合、サーバーの衝突が発生する場所は、ネットが分割する必要がある場所を示しています。

5.4 サーバークライアント接続の終了

クライアント接続が予期せず閉じると、クライアントが接続されているサーバーによってクライアントに代わって終了メッセージが生成されます。他のメッセージは生成または使用されることはありません。

5.5 サーバーサーバー接続の終了

Squitコマンドまたは「自然」の原因のいずれかを介してサーバーサーバー接続が閉じている場合、接続されたIRCネットワークの残りの部分は、閉鎖を検出したサーバーによって情報を更新する必要があります。終了サーバーは、スクイットのリストを送信します（その接続の背後にあるサーバーごとに1つ）。（セクション4.1.6（squit）を参照）。

5.6 ニックネームの変更を追跡します

すべてのIRCサーバーは、最近のニックネームの変更の履歴を維持するために必要です。これは、ニック変更のレース条件がそれらを操作するコマンドで発生したときに、サーバーが物事に接触する機会を持つことができるために重要です。ニックの変更を追跡する必要があるメッセージは次のとおりです。

* キル（ニックが切断されている）

* モード（ / - o、vチャンネル）

* キック（ニックがチャンネルから削除されている）

ニックの変更を確認する必要はありません。

上記の場合、サーバーは最初にニックネームの存在を確認し、次にその履歴を確認して、そのニックが現在属している人を確認する必要があります（誰かがいれば！）。これにより、レース条件の可能性が減りますが、サーバーが間違ったクライアントに影響を与えることになっている場合でも発生する可能性が

ending up affecting the wrong client. When performing a change trace for an above command it is RECOMMENDED that a time range be given and entries which are too old ignored.

For a reasonable history, a server SHOULD be able to keep previous nickname for every client it knows about if they all decided to change. This size is limited by other factors (such as memory, etc).

5.7 Tracking recently used nicknames

This mechanism is commonly known as "Nickname Delay", it has been proven to significantly reduce the number of nickname collisions resulting from "network splits"/reconnections as well as abuse.

In addition of keeping track of nickname changes, servers SHOULD keep track of nicknames which were recently used and were released as the result of a "network split" or a KILL message. These nicknames are then unavailable to the server local clients and cannot be re-used (even though they are not currently in use) for a certain period of time.

The duration for which a nickname remains unavailable SHOULD be set considering many factors among which are the size (user wise) of the IRC network, and the usual duration of "network splits". It SHOULD be uniform on all servers for a given IRC network.

5.8 Flood control of clients

With a large network of interconnected IRC servers, it is quite easy for any single client attached to the network to supply a continuous stream of messages that result in not only flooding the network, but also degrading the level of service provided to others. Rather than require every 'victim' to provide their own protection, flood protection was written into the server and is applied to all clients except services. The current algorithm is as follows:

* check to see if client's `message timer' is less than current time (set to be equal if it is);

* read any data present from the client;

* while the timer is less than ten (10) seconds ahead of the current time, parse any present messages and penalize the client by two (2) seconds for each message;

* additional penalties MAY be used for specific commands which generate a lot of traffic across the network.

This in essence means that the client may send one (1) message every two (2) seconds without being adversely affected. Services MAY also be subject to this mechanism.

5.9 Non-blocking lookups

In a real-time environment, it is essential that a server process does as little waiting as possible so that all the clients are serviced fairly. Obviously this requires non-blocking IO on all network read/write operations. For normal server connections, this was not difficult, but there are other support operations that may cause the server to block (such as disk reads). Where possible, such activity SHOULD be performed with a short timeout.

あります。上記のコマンドの変更トレースを実行する場合、時間範囲を指定し、古すぎるエントリを無視することをお勧めします。

合理的な履歴のために、サーバーはすべてのクライアントの以前のニックネームを保持できるはずです。このサイズは、他の要因（メモリなど）によって制限されます。

5.7 追跡最近使用されたニックネーム

このメカニズムは一般に「ニックネーム遅延」として知られており、「ネットワークスプリット」/再接続と乱用に起因するニックネーム衝突の数を大幅に減らすことが証明されています。

ニックネームの変更を追跡することに加えて、サーバーは最近使用され、「ネットワーク分割」またはキルメッセージの結果としてリリースされたニックネームを追跡する必要があります。これらのニックネームは、サーバーのローカルクライアントが利用できず、一定期間（現在使用されていない場合でも）再利用することはできません。

ニックネームが利用できないままである期間は、IRCネットワークのサイズ（ユーザーワイズ）、および「ネットワークスプリット」の通常の期間である多くの要因を考慮して設定する必要があります。特定のIRCネットワークのすべてのサーバーで均一でなければなりません。

5.8 クライアントの洪水制御

相互接続されたIRCサーバーの大きなネットワークを使用すると、ネットワークに接続されている単一のクライアントが、ネットワークにあふれているだけでなく、他のサービスのレベルを低下させるメッセージの連続ストリームを提供することが非常に簡単です。すべての「被害者」に独自の保護を提供することを要求するのではなく、洪水保護はサーバーに書き込まれ、サービスを除くすべてのクライアントに適用されます。現在のアルゴリズムは次のとおりです。

* クライアントの「メッセージタイマー」が現在の時刻よりも小さいかどうかを確認してください（そうであれば等しく設定）。

* クライアントから存在するデータをお読みください。

* タイマーは現在の時刻より10秒未満ですが、現在のメッセージを解析し、各メッセージに対してクライアントに2秒秒をペナルティします。

* ネットワーク全体で多くのトラフィックを生成する特定のコマンドに追加の罰則が使用される場合があります。

これは本質的に、クライアントが悪影響を受けずに2秒ごとに1つのメッセージを送信できることを意味します。サービスは、このメカニズムの対象となる場合があります。

5.9 非ブロッキング検索

リアルタイム環境では、すべてのクライアントが公正にサービスされるように、サーバープロセスができるだけ待機しないことが不可欠です。明らかに、これにはすべてのネットワーク読み取り/書き込み操作で非ブロッキングIOが必要です。通常のサーバー接続の場合、これは難しくありませんでしたが、サーバーがブロックされる可能性のある他のサポート操作があります（ディスク読み取りなど）。可能であれば、そのようなアクティビティは短いタイムアウトで実行する必要があります。

5.9.1 Hostname (DNS) lookups

Using the standard resolver libraries from Berkeley and others has meant large delays in some cases where replies have timed out. To avoid this, a separate set of DNS routines were written for the current implementation. Routines were setup for non-blocking IO operations with local cache, and then polled from within the main server IO loop.

5.9.2 Username (Ident) lookups

Although there are numerous ident libraries (implementing the "Identification Protocol" [IDENT]) for use and inclusion into other programs, these caused problems since they operated in a synchronous manner and resulted in frequent delays. Again the solution was to write a set of routines which would cooperate with the rest of the server and work using non-blocking IO.

6. Current problems

There are a number of recognized problems with this protocol, all of which are hoped to be solved sometime in the near future during its rewrite. Currently, work is underway to find working solutions to these problems.

6.1 Scalability

It is widely recognized that this protocol does not scale sufficiently well when used in a large arena. The main problem comes from the requirement that all servers know about all other servers and clients and that information regarding them be updated as soon as it changes. It is also desirable to keep the number of servers low so that the path length between any two points is kept minimal and the spanning tree as strongly branched as possible.

6.2 Labels

The current IRC protocol has 4 types of labels: the nickname, the channel name, the server name and the service name. Each of the four types has its own domain and no duplicates are allowed inside that domain. Currently, it is possible for users to pick the label for any of the first three, resulting in collisions. It is widely recognized that this needs reworking, with a plan for unique names for nicks that don't collide being desirable as well as a solution allowing a cyclic tree.

6.2.1 Nicknames

The idea of the nickname on IRC is very convenient for users to use when talking to each other outside of a channel, but there is only a finite nickname space and being what they are, it's not uncommon for several people to want to use the same nick. If a nickname is chosen by two people using this protocol, either one will not succeed or both will be removed by use of KILL (See Section 3.7.1 of "IRC Client Protocol" [IRC-CLIENT]).

6.2.2 Channels

The current channel layout requires that all servers know about all channels, their inhabitants and properties. Besides not scaling well, the issue of privacy is also a concern. A collision of channels is treated as an inclusive event (people from both nets

5.9.1 ホスト名（DNS）ルックアップ

バークレーなどの標準のリゾルバーライブラリを使用すると、返信がタイムアウトした場合には大きな遅延があります。これを回避するために、DNSルーチンの個別のセットが現在の実装のために記述されました。ルーチンは、ローカルキャッシュを使用した非ブロッキングIO操作用にセットアップされ、メインサーバーIOループ内からポーリングされました。

5.9.2 ユーザー名（識別）ルックアップ

他のプログラムに使用および包含するための多くの識別ライブラリ（「識別プロトコル」[識別] [識別]を実装）がありますが、これらは同期的に動作し、頻繁な遅延をもたらすため問題を引き起こしました。繰り返しますが、ソリューションは、サーバーの残りの部分と協力し、非ブロッキングIOを使用して作業するルーチンのセットを作成することでした。

6. 現在の問題

このプロトコルには多くの認識された問題があり、そのすべてが、その書き直し中に近い将来に解決されることを望んでいます。現在、これらの問題の実用的な解決策を見つけるための作業が進行中です。

6.1 スケーラビリティ

このプロトコルは、大きなアリーナで使用した場合、十分に十分に拡大しないことが広く認識されています。主な問題は、すべてのサーバーが他のすべてのサーバーとクライアントについて知っているという要件と、それらに関する情報が変更されるとすぐに更新されることから生じています。また、サーバーの数を低く抑えて、任意の2つのポイント間のパス長が最小限に抑えられ、スパニングツリーができるだけ強く分岐するようにすることも望ましいです。

6.2 ラベル

現在のIRCプロトコルには、ニックネーム、チャネル名、サーバー名、サービス名の4種類のラベルがあります。4つのタイプにはそれぞれ独自のドメインがあり、そのドメイン内で重複は許可されていません。現在、ユーザーは最初の3つのいずれかでラベルを選択して衝突を起こす可能性があります。これには再加工が必要であり、衝突しないニックのユニークな名前の計画と、循環ツリーを許可するソリューションの計画が広く認識されています。

6.2.1 ニックネーム

IRCのニックネームのアイデアは、ユーザーがチャンネルの外でお互いに話し合うときに使用するのに非常に便利ですが、有限のニックネームスペースしかあり、それらが何であるかだけで、何人かが同じを使いたいと思うことは珍しくありませんニック。このプロトコルを使用して2人によってニックネームが選択されている場合、キルを使用して成功しないか、その両方が削除されます（「IRCクライアントプロトコル」[IRC-Client]のセクション3.7.1を参照）。

6.2.2 チャネル

現在のチャネルレイアウトでは、すべてのサーバーがすべてのチャネル、その住民、およびプロパティについて知っていることが必要です。うまくスケーリングしないことに加えて、プライバシーの問題も懸念事項です。チャネルの衝突は、ニックネー

on channel with common name are considered to be members of it) rather than an exclusive one such as used to solve nickname collisions.

This protocol defines "Safe Channels" which are very unlikely to be the subject of a channel collision. Other channel types are kept for backward compatibility.

6.2.3 Servers

Although the number of servers is usually small relative to the number of users and channels, they too are currently REQUIRED to be known globally, either each one separately or hidden behind a mask.

6.3 Algorithms

In some places within the server code, it has not been possible to avoid N^2 algorithms such as checking the channel list of a set of clients.

In current server versions, there are only few database consistency checks, most of the time each server assumes that a neighbouring server is correct. This opens the door to large problems if a connecting server is buggy or otherwise tries to introduce contradictions to the existing net.

Currently, because of the lack of unique internal and global labels, there are a multitude of race conditions that exist. These race conditions generally arise from the problem of it taking time for messages to traverse and effect the IRC network. Even by changing to unique labels, there are problems with channel-related commands being disrupted.

7. Security Considerations

7.1 Authentication

Servers only have two means of authenticating incoming connections: plain text password, and DNS lookups. While these methods are weak and widely recognized as unsafe, their combination has proven to be sufficient in the past:

* public networks typically allow user connections with only few restrictions, without requiring accurate authentication.

* private networks which operate in a controlled environment often use home-grown authentication mechanisms not available on the internet: reliable ident servers [IDENT], or other proprietary mechanisms.

The same comments apply to the authentication of IRC Operators.

It should also be noted that while there has been no real demand over the years for stronger authentication, and no real effort to provide better means to safely authenticate users, the current protocol offers enough to be able to easily plug-in external authentication methods based on the information that a client can submit to the server upon connection: nickname, username, password.

7.2 Integrity

Since the PASS and OPER messages of the IRC protocol are sent in clear text, a stream layer encryption mechanism (like "The TLS Protocol" [TLS]) could be used to protect these transactions.

ムの衝突を解決するために使用されるような排他的なものではなく、包括的なイベントとして扱われます（共通名のチャネル上の両方の網の人々はメンバーと見なされます）。

このプロトコルは、チャネル衝突の対象となる可能性が非常に低い「安全なチャネル」を定義します。他のチャネルタイプは、後方互換性のために保持されます。

6.2.3 サーバー

サーバーの数は通常、ユーザーとチャネルの数に比べて小さいですが、それらも現在、それぞれが個別にまたはマスクの後ろに隠れているかのいずれかをグローバルに知っている必要があります。

6.3 アルゴリズム

サーバーコード内の一部の場所では、クライアントのセットのチャネルリストをチェックするなど、n^2アルゴリズムを回避することはできませんでした。

現在のサーバーバージョンでは、データベースの一貫性チェックはほとんどありません。ほとんどの場合、各サーバーは隣接するサーバーが正しいと想定しています。これにより、接続サーバーがバギーであるか、そうでなければ既存のネットと矛盾を導入しようとする場合、大きな問題への扉が開かれます。

現在、ユニークな内部およびグローバルラベルが不足しているため、存在する多くの人種条件があります。これらの人種条件は、一般に、メッセージがIRCネットワークを通過して影響を与えるのに時間がかかるという問題から生じます。一意のラベルに変更しても、チャネル関連のコマンドが破壊されていることに問題があります。

7. セキュリティに関する考慮事項

7.1 認証

サーバーには、着信接続を認証する2つの手段しかありません。プレーンテキストパスワードとDNSルックアップです。これらの方法は弱く、安全でないと広く認識されていますが、それらの組み合わせは過去に十分であることが証明されています。

* 公開ネットワークは通常、正確な認証を必要とせずに、制限が少ないユーザー接続を許可します。

* 制御された環境で動作するプライベートネットワークは、多くの場合、インターネット上で利用できない自家製認証メカニズムを使用します。信頼できる識別サーバー[識別]、またはその他の独自のメカニズム。

同じコメントは、IRCオペレーターの認証に適用されます。

また、より強力な認証のために長年にわたって実際の需要はありませんでしたが、ユーザーを安全に認証するためのより良い手段を提供するための実際の努力はありませんが、現在のプロトコルは、外部認証メソッドを簡単にプラグインすることができるほど十分に提供しますクライアントが接続時にサーバーに送信できる情報について：ニックネーム、ユーザー名、パスワード。

7.2 威厳

IRCプロトコルのパスおよびオペレーションメッセージはクリアテキストで送信されるため、これらのトランザクションを保護するために、ストリームレイヤー暗号化メカニズム（「TLSプロトコル」[TLS]など）を使用できます。

8. Current support and availability

```
Mailing lists for IRC related discussion:
  General discussion: ircd-users@irc.org
  Protocol development: ircd-dev@irc.org
```

```
Software implementations:
  ftp://ftp.irc.org/irc/server
  ftp://ftp.funet.fi/pub/unix/irc
  ftp://coombs.anu.edu.au/pub/irc
```

Newsgroup: alt.irc

8. 現在のサポートと可用性

NewsGroup：Alt.Irc

9. Acknowledgements

Parts of this document were copied from the RFC 1459 [IRC] which first formally documented the IRC Protocol. It has also benefited from many rounds of review and comments. In particular, the following people have made significant contributions to this document:

Matthew Green, Michael Neumayer, Volker Paulsen, Kurt Roeckx, Vesa Ruokonen, Magnus Tjernstrom, Stefan Zehl.

9. 謝辞

このドキュメントの一部は、最初にIRCプロトコルを正式に文書化したRFC 1459 [IRC]からコピーされました。また、多くのラウンドのレビューとコメントの恩恵を受けています。特に、次の人々がこの文書に多大な貢献をしています。

マシュー・グリーン、マイケル・ノイマイヤー、ヴォルカー・ポールセン、カート・ロックズ、ヴェサ・ルーコネン、マグナス・ツェーンストロム、ステファン・ゼール。

10. References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[IRC] Oikarinen, J. and D. Reed, "Internet Relay Chat Protocol", RFC 1459, May 1993.

[IRC-ARCH] Kalt, C., "Internet Relay Chat: Architecture", RFC 2810, April 2000.

[IRC-CLIENT] Kalt, C., "Internet Relay Chat: Client Protocol", RFC 2812, April 2000.

[IRC-CHAN] Kalt, C., "Internet Relay Chat: Channel Management", RFC 2811, April 2000.

[ZLIB] Deutsch, P. and J-L. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.

[DEFLATE] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

[GZIP] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, May 1996.

[IDENT] St. Johns, M., "The Identification Protocol", RFC 1413, February 1993.

[TLS] Dierks, T. and C. Allen, "The TLS Protocol", RFC 2246, January 1999.

10. 参考文献

[キーワード] Bradner、S。、「要件レベルを示すためにRFCで使用するためのキーワード」、BCP 14、RFC 2119、1997年3月。

[ABNF] Crocker、D。およびP. Overell、「構文仕様のためのBNFの増強：ABNF」、RFC 2234、1997年11月。

[IRC] Oikarinen、J。およびD. Reed、「インターネットリレーチャットプロトコル」、RFC 1459、1993年5月。

[IRC-Arch] Kalt、C。、「インターネットリレーチャット：アーキテクチャ」、RFC 2810、2000年4月。

[IRC-Client] Kalt、C。、「インターネットリレーチャット：クライアントプロトコル」、RFC 2812、2000年4月。

[IRC-chan] Kalt、C。、「インターネットリレーチャット：チャネル管理」、RFC 2811、2000年4月。

[Zlib] Deutsch、P。およびJ-L。Gailly、「Zlib圧縮データ形式の仕様バージョン3.3」、RFC 1950、1996年5月。

[Deflate] Deutsch、P。、「Deflate圧縮データ形式仕様バージョン1.3」、RFC 1951、1996年5月。

[GZIP] Deutsch、P。、「GZIPファイル形式の仕様バージョン4.3」、RFC 1952、1996年5月。

[識別]セントジョンズ、M。、「識別プロトコル」、RFC 1413、1993年2月。

[TLS] Dierks、T。およびC. Allen、「TLSプロトコル」、RFC 2246、1999年1月。

11. Author's Address

Christophe Kalt 99 Teaneck Rd, Apt #117 Ridgefield Park, NJ 07660 USA

```
EMail: kalt@stealth.net
```

11. 著者の連絡先

Christophe Kalt 99 Teaneck Rd、Apt # 117 Ridgefield Park、NJ 07660 USA

12. Full Copyright Statement

12. 完全な著作権声明

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

謝辞

RFCエディター機能の資金は現在、インターネット協会によって提供されています。