```
Network Working Group                                       M. Mikulenas
Internet-Draft                                               Independent
Updates: 1459, 2812 (if approved)                              D. Oakley
Intended status: Informational                                  ircdocs
Expires: November 30, 2017                                  May 29, 2017
```

Internet Relay Chat: Client-to-Client Protocol (CTCP)
draft-oakley-irc-ctcp-00

Abstract

This document describes the Client-to-Client Protocol (CTCP), which lets Internet Relay Chat (IRC) clients send each other messages that get displayed or responded to in special ways. CTCP has been widely implemented, with most clients supporting it natively. This document outlines how to implement CTCP and the most common messages used.

It updates RFC 1459 and RFC 2812.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The core Internet Relay Chat (IRC) protocol as described in [RFC1459] and [RFC2812] only has a single command for regular user messages, and does not provide a way for clients to exchange information directly. Client-to-Client Protocol (CTCP) messages let clients exchange messages that get displayed or responded to in special ways. Some examples of how CTCP is used is to request special formatting on messages, query other clients for metadata, and help initiate file transfers with other clients.

This document goes over the subset of CTCP which is commonly implemented, and is compatible with clients implementing CTCP as described by older documents.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Message Syntax

CTCP queries are sent with the PRIVMSG IRC command, and CTCP replies are sent with NOTICE command. To indicate a CTCP query or reply, the body of the message (the second parameter) begins with the CTCP delimiter. The ABNF [RFC5234] for this message body is as follows:

```
     delim    = %x01

     command  = 1*( %x02-09 / %x0B-0C / %x0E-1F / %x21-FF )
                   ; any octet except NUL, delim, CR, LF, and " "

     params   = 1*( %x02-09 / %x0B-0C / %x0E-FF )
                   ; any octet except NUL, delim, CR, and LF

     body     = delim command [ SPACE params ] [ delim ]
```

Commands are case-insensitive. When creating new CTCP commands, authors SHOULD use only alphanumeric characters for ease of implementation.

The final CTCP delimiter SHOULD be sent on outgoing messages for compatibility reasons, and software MUST accept incoming messages which lack it. This is due to how certain clients implement message splitting and truncation.

Queries MAY be sent to channels. When these queries are responded to, the responses are sent to the querying client rather than the channel which the query was originally sent to.

Many servers implement optional filtering so that only the ACTION CTCP message can be sent to channels. When this is done, it can typically be enabled or disabled by channel operators with a channel mode.

Here are two examples of CTCP queries and replies:

```
  :alice PRIVMSG bob :\x01VERSION\x01
  :bob NOTICE alice :\x01VERSION Snak for Macintosh 4.13 English\x01

  :alice PRIVMSG #ircv3 :\x01PING 1473523796 918320\x01
  :bob NOTICE alice :\x01PING 1473523796 918320\x01
```

3. Message Types

CTCP messages generally take on one of these types. These message types are defined here to simplify understanding, and aren't differentiated by the protocol itself.

3.1. Extended formatting

This type of CTCP is used to request special formatting of a user-visible message. That is, to send a user-visible message that should be displayed differently from regular messages - e.g. as an action, a whisper, an announcement.

Extended formatting messages are sent as a PRIVMSG, and are expected to be used in channels as well as between clients. There is no automatic response to this message type, as it is not a query nor reply.

These messages are sent as a PRIVMSG and can have parameters, but generate no reply.

Example:

```
  :dan PRIVMSG #ircv3 :\x01ACTION writes some specs!\x01
```

3.2. Metadata Query

This type of CTCP is used to provide relatively static information about the target client, user or connection.

This CTCP takes the form of a query and a response (as a PRIVMSG and NOTICE, respectively). Due to how bouncers interact with multiple clients, there may sometimes be multiple responses to queries.

Metadata queries MUST NOT require the recipient to implement any side effects (beyond sending the reply itself); if a CTCP message causes side effects by design, it should be categorized as an extended query instead.

Metadata queries do not have any parameters, but expect a reply with parameters as the response data.

Example:

```
  :alice PRIVMSG bob :\x01VERSION\x01
  :bob NOTICE alice :\x01VERSION SaberChat 27.5\x01
```

3.3. Extended Query

This type of CTCP is used to provide dynamic information or invoke actions from the client.

This CTCP takes the form of a query and a response (as a PRIVMSG and NOTICE, respectively).

Queries sent to a channel always generate private replies.

Extended queries and replies may have parameters.

Example:

```
  :alice PRIVMSG bob :\x01PING 1473523796 918320\x01
  :bob NOTICE alice :\x01PING 1473523796 918320\x01
```

4. Messages

CTCP messages themselves are not standardised. Clients that receive either unexpected messages or known messages with unexpected values SHOULD ignore them and produce no response to the sending user. Clients MAY receive more than one response per user for a query they send, due to multiple clients being connected behind an IRC bouncer.

5. Acknowledgements

Thanks to the IRCv3 group for giving feedback on this specification, and to Khaled for advice on client flood protection.

Thanks to Michael Sandrof for creating CTCP, Troy Rollo for creating the related DCC protocol, as well as Klaus Zeuge and Ben Mesander who wrote and revised related specifications.

Special thanks to dequis, Peter Powell and James Wheare for help with this and related work.

6. Security Considerations

CTCP messages are completely untrusted data, and clients MUST NOT assume that they are well-formed or complete.

Older CTCP specifications describe quoting methods which are complex and not widely implemented. Implementations SHOULD NOT implement "low-level quoting" or "CTCP-level quoting" when parsing messages.

Older CTCP specifications describe including more than one CTCP message inside a single PRIVMSG or NOTICE command. Implementations SHOULD NOT implement this form of CTCP parsing as it is not widely-implemented and may result in an implementation that can be more easily flooded off the server they are connected to.

CTCP requests can be abused to flood clients off the server they are connected to. Clients may ignore or delay excessive incoming requests to protect against this.

7. IANA Considerations

This document has no actions for IANA.

8. Normative References

[RFC1459] Oikarinen, J. and D. Reed, "Internet Relay Chat Protocol", RFC 1459, DOI 10.17487/RFC1459, May 1993, <http://www.rfc-editor.org/info/rfc1459>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2812] Kalt, C., "Internet Relay Chat: Client Protocol", RFC 2812, DOI 10.17487/RFC2812, April 2000, <http://www.rfc-editor.org/info/rfc2812>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <http://www.rfc-editor.org/info/rfc5322>.

Appendix A. Message List

This section is not exhaustive, and only lists those CTCP messages which are widely implemented across the IRC ecosystem.

The reply and parameter lines below use a simplified syntax that represents variables by surrounding them with angle brackets,.

A.1. ACTION

```
Type:    Extended Formatting
Params:  ACTION <text>
```

This extended formatting message shows that <text> should be displayed as a third-person action or emote; in clients, it's generally activated with the command "/me".

ACTION is universally implemented and very commonly used. Clients MUST implement this CTCP message to effectively use IRC.

Example:

```
  Raw:        :dan!user@host PRIVMSG #ircv3 :\x01ACTION does it!\x01

  Formatted:  * dan does it!
```

A.2. CLIENTINFO

```
Type:   Extended Query
Reply:  CLIENTINFO <tokens>
```

This extended query returns a list of the CTCP messages that this client supports and implements, delimited by a single ASCII space.

CLIENTINFO is widely implemented. Clients SHOULD implement this CTCP message.

Example:

```
  Query:     CLIENTINFO
  Response:  CLIENTINFO ACTION DCC CLIENTINFO PING TIME VERSION
```

A.3. DCC

```
Type:    Extended Query
Params:  DCC <type> <argument> <host> <port>
```

This extended query is used to setup and control connections that go directly between clients, bypassing the IRC server. This is typically used for features that require a large amount of traffic between clients or simply wish to bypass the server itself such as file transfer and direct chat.

The Direct Client-to-Client (DCC) Protocol requires its own specification, and is not described in-depth here.

DCC is widely implemented. Clients MAY implement this CTCP message.

A.4. FINGER

```
Type:   Metadata Query
Reply:  FINGER <info>
```

This metadata query returns miscellaneous info about the user, typically the same information that's held in their realname field.

However, some implementations return the client name and version instead.

FINGER is largely obsolete. Clients MAY implement this CTCP message.

Example:

```
  Query:     FINGER
  Response:  FINGER WeeChat 1.8-dev
```

A.5. PING

```
Type:    Extended Query
Params:  PING <info>
```

This extended query is used to confirm reachability with other clients and to check latency. When receiving a CTCP PING, the reply MUST contain exactly the same parameters as the original query.

PING is universally implemented. Clients SHOULD implement this CTCP message.

Example:

```
  Query:     PING 1473523721 662865
  Response:  PING 1473523721 662865

  Query:     PING foo bar baz
  Response:  PING foo bar baz
```

A.6. SOURCE

```
Type:   Metadata Query
Reply:  SOURCE <info>
```

This metadata query is used to return the location of the source code for the client.

SOURCE is rarely implemented. Clients MAY implement this CTCP message.

Example:

```
  Query:     SOURCE
  Response:  SOURCE https://weechat.org/download
```

A.7. TIME

```
Type:    Extended Query
Params:  TIME <timestring>
```

This extended query is used to return the client's local time in an unspecified human-readable format. In practice, both the format output by ctime() and the format described in Section 3.3 of [RFC5322] are common.

New implementations MAY default to UTC time for privacy reasons.

TIME is almost universally implemented. Clients SHOULD implement this CTCP message.

Example:

```
  Query:     TIME
  Response:  TIME Mon, 08 May 2017 09:15:29 GMT
```

A.8. VERSION

Type: Metadata Query
Reply: VERSION <verstring>

This metadata query is used to return the name and version of the client software in use. There is no specified format for the version string.

Clients may allow users to customise the response value for this query.

VERSION is universally implemented. Clients SHOULD implement this CTCP message.

Example:

```
  Query:     VERSION
  Response:  VERSION WeeChat 1.8-dev (git: v1.7-329-g22f2fd03a)
```

A.9. USERINFO

Type: Metadata Query
Reply: USERINFO <info>

This metadata query returns miscellaneous info about the user, typically the same information that's held in their realname field.

However, some implementations return "<nickname> (<realname>)" instead.

USERINFO is largely obsolete. Clients MAY implement this CTCP message.

Example:

```
Query:     USERINFO
Response:  USERINFO fred (Fred Foobar)
```

Authors' Addresses

Mantas Mikulenas
Independent

Email: grawity@gmail.com

Daniel Oakley
ircdocs

Email: daniel@danieloaks.net