Network Working Group
Internet-Draft
Expires: July 2, 2005

L. Hardy
E. Brocklesby
ircd-ratbox
K. Mitchell
Undernet IRC Network
January 2005

IRC RPL_ISUPPORT Numeric Definition
draft-hardy-irc-isupport-00

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she become aware will be disclosed, in accordance with RFC 3668.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on July 2, 2005.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

IRC (Internet Relay Chat) is a long-standing protocol for real-time chatting. Servers often provide features that are backward compatible with older clients, but do not provide a reliable method for making clients aware of what extensions exist. This memo presents a method for servers to advertise such extensions to

clients.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

Table of Contents

1. Introduction

The IRC protocol, as originally documented by RFC 1459 [2] and updated by RFC 2812 [3], is a simple, text-based conferencing protocol, involving a number of users spread across a number of interconnected servers. These users may chat with other individual users, or may chat with groups of users on "channels"--what other chat systems refer to as "rooms" or "chat rooms". These channels have various "modes" which can alter the behaviour of the channel.

Over the years, various modifications to the basic IRC protocol have been made by IRC server programmers. These modifications may amongst other things, introduce features available to users and remove some features that were available to users. Due to the wildly varying nature between IRC servers of what features are supported and how the protocol differs from RFC 1459 [2] and RFC 2119 [1], it is problematic for clients to determine just how a server differs from these specifications.

A de facto standard emerged in the community, originally implemented by the Undernet's IRC server software based on the 005 numeric from DALnet's IRC server. This standard allows the server to advertise to the client upon connection which extensions to the protocol are supported. This reply, termed RPL_ISUPPORT, uses the non-standard numeric 005.

Unfortunately, whilst the numeric itself has become a de-facto standard, differences have emerged between the meaning of extensions advertised. This memo attempts to standardise the format of the RPL_ISUPPORT numeric, as well as standardising the definition of extensions given within this numeric.

2. Problems to be Solved

The IRC protocol is no longer standardised in many aspects. This means that various IRC daemons support features others do not, and clients need a reliable method of determining which features a server supports. The de-facto standard that emerged to counter this has itself suffered from a lack of standards.

The solution to these problems is to formally define a method for servers to advertise to clients the features it supports, and to formally define the features advertised. The client may then rely on this method for reliably determining what features a server supports.

3. The RPL_ISUPPORT numeric

The extension for informing clients about available features is implemented by the addition of one numeric, RPL_ISUPPORT, which contains a list of features supported by the server. Clients SHOULD NOT assume a server supports a feature unless it has been advertised in RPL_ISUPPORT.

In ABNF [4] notation:

```
       isupport     =  [ ":" servername SP ] "005" SP nick SP
                       1*13( token SP ) ":are supported by this server"

       token        = *1"-" parameter / parameter *1( "=" value )
       parameter    = 1*20letter
       value        = *letpun
       letter       = ALPHA / DIGIT
       punct        = %d33-47 / %d58-64 / %d91-96 / %d123-126
       letpun       = letter / punct
```

where SP is as designated in Appendix A of RFC 2234 [4], and servername and nick are as designated in section 2.3.1 of RFC 1459 [2]. As with other local numerics, when RPL_ISUPPORT is delivered remotely, it MUST be converted into a "105" numeric before delivery to the client.

A token is of the form "-PARAMETER", "PARAMETER" or "PARAMETER=VALUE". A server MAY send an empty value field and a parameter MAY have a default value. A server MUST send the parameter in upper-case text. Unless otherwise stated, when a parameter contains a value, the value MUST be treated as being case sensitive. The value MAY contain multiple fields, if this is the case the fields MUST be delimited with a comma character (',').

Due to the increasingly modular nature of IRC server software, it is possible that the status of features previously advertised to clients in RPL_ISUPPORT can change. When this happens, a server SHOULD reissue the RPL_ISUPPORT numeric with the relevant parameters that have changed. If a feature becomes unavailable, the server MUST prefix the parameter with the dash character ('-') when issuing the updated RPL_ISUPPORT.

As RFC 1459 [2] limits the maximum parameters of any reply to 15, the maximum amount of extensions that may be advertised is 13. To counter this, a server MAY issue multiple RPL_ISUPPORT numerics. A server MUST issue the RPL_ISUPPORT numeric after client registration has completed. It MUST be issued after the RPL_WELCOME (XXX - reference?) welcome and MUST be issued before any further commands from the client are processed.

4. Parameters

Servers MAY send parameters that are not covered in this specification.

4.1 CASEMAPPING

CASEMAPPING=string

The CASEMAPPING parameter is used to indicate what method is used by the server to compare equality of case-insensitive strings. Possible values are:

- o "ascii": The ASCII characters 97 to 122 (decimal) are defined as the lower-case characters of ASCII 65 to 90 (decimal). No other character equivalency is defined.

- o "rfc1459": The ASCII characters 97 to 126 (decimal) are defined as the lower-case characters of ASCII 65 to 94 (decimal). No other character equivalency is defined.

- o "strict-rfc1459": The ASCII characters 97 to 125 (decimal) are defined as the lower-case characters of ASCII 65 to 93 (decimal). no other character equivalency is defined.

The value MUST be specified. Whilst RFC 1459 [2] defines character equivalency in terms of "strict-rfc1459" encoding, this is believed to be a mistake as the majority of IRC server implementations treat character equivalency in terms of "rfc1459" encoding with the tilde character ('~') and caret character ('^') being equivalent.

An example CASEMAPPING token:
CASEMAPPING=rfc1459

4.2 CHANLIMIT

CHANLIMIT=prefix:number[,prefix:number[,...]]

The CHANLIMIT parameter is used to indicate the maximum amount of channels that a client may join. The value is a series of "prefix:number" pairs, where "prefix" refers to one or more prefix characters defined in the PREFIX (Section 4.15) token, and "number" indicates how many channels of the given type combined may be joined. The number parameter MAY be omitted if no limit is placed on the number of channels.

A client SHOULD NOT make any assumptions about how many channels other clients may join based on the CHANLIMIT parameter.

An example CHANLIMIT token:
CHANLIMIT=#+:25,&:
Indicates that a client may join up to 25 channels with the prefix '#' and '+', and an unlimited amount of channels with the '&' prefix.

4.3 CHANMODES

CHANMODES=A,B,C,D

The CHANMODES parameter is used to indicate the channel modes available and the arguments they take. There are four categories of modes, defined as follows:

- o Type A: Modes that add or remove an address to or from a list. These modes MUST always have a parameter when sent from the server to a client. A client MAY issue the mode without an argument to obtain the current contents of the list.

- o Type B: Modes that change a setting on a channel. These modes MUST always have a parameter.

- o Type C: Modes that change a setting on a channel. These modes MUST have a parameter when being set, and MUST NOT have a parameter when being unset.

- o Type D: Modes that change a setting on a channel. These modes MUST NOT have a parameter.

To allow for future extensions, a server MAY send additional types, delimeted by the comma character (','). The behaviour of any additional types is undefined.

The IRC server MUST NOT list modes in the CHANMODES parameter that are contained within the PREFIX (Section 4.15) parameter. However, for completeness, modes within the PREFIX (Section 4.15) parameter may be treated as type B modes.

An example CHANMODES token:
CHANMODES=b,k,l,imnpst

4.4 CHANNELLEN

CHANNELLEN=number

The CHANNELLEN parameter specifies the maximum length of a channel name that a client may join. A client elsewhere on the network MAY join a channel with a name length of higher value. The value MUST be specified and MUST be numeric.

An example CHANNELLEN token:
CHANNELLEN=50
Limits the length of a channel name that a user may join to 50 characters.

4.5 CHANTYPES

CHANTYPES=[string]

Special characters used as prefixes are reserved to differentiate channels from other namespaces within the IRC protocol. The CHANTYPES parameter specifies these characters.

The value is OPTIONAL and when is not specified indicates that no channel types are supported.

An example CHANTYPES token:
CHANTYPES=&#
Denotes the ampersand ('&') and hash ('#') characters as channel prefixes.

4.6 CNOTICE

CNOTICE

The CNOTICE parameter indicates that the server supports the "CNOTICE" command. An extension for the NOTICE command, as defined in RFC 1459 [2] section 4.4.2, it allows users with a specific status in a channel to issue a NOTICE command to a user within that channel, free of certain restrictions a server MAY apply to NOTICE.

The CNOTICE parameter MUST NOT be specified with a value.

An example CNOTICE token:
CNOTICE

4.7 CPRIVMSG

CPRIVMSG

The CPRIVMSG parameter indicates that the server supports the "CPRIVMSG" command. An extension for the PRIVMSG command, as defined in RFC 1459 [2] section 4.4.1, it allows users with a specific status in a channel to issue a PRIVMSG command to a user within that channel, free of certain restrictions a server MAY apply to PRIVMSG.

The CPRIVMSG parameter MUST NOT be specified with a value.

An example CPRIVMSG token:
CPRIVMSG

4.8 ELIST

ELIST=string

The ELIST parameter indicates that the server supports search extensions to the LIST command. The value is required, and is a non-delimited set of letters which each denote an extension. The following extensions, which a client MUST treat as being case insensitive are defined:

- o C: Searching based on creation time, via the "C<val" and "C>val" modifiers to search for a channel creation time that is lower or higher than val respectively.
- o M: Searching based on mask.
- o N: Searching based on !mask.
- o P: To explain
- o T: Searching based on topic time, via the "T<val" and "T>val" modifiers to search for a topic time that is lower or higher than val respectively.
- o U: Searching based on user count within the channel, via the "<val" and ">val" modifiers to search for a channel that has less than or more than val users respectively.

An example ELIST token:
ELIST=CMNTU

4.9 EXCEPTS

EXCEPTS[=letter]

The EXCEPTS parameter indicates that the server supports "ban exceptions", as defined in RFC 2811 [5] section 4.3.1. The value is OPTIONAL and when is not specified indicates that that the letter 'e' is used as the channel mode for ban exceptions. When the value is specified, it indicates the letter which is used for ban exceptions.

An example EXCEPTS token:
EXCEPTS

4.10 INVEX

INVEX[=letter]

The INVEX parameter indicates that the server supports "invite exceptions", as defined in RFC 2811 [5] section 4.3.2. The value is OPTIONAL and when is not specified indicates that the letter 'I' is used as the channel mode for invite exceptions. When the value is specified, it indicates the letter which is used for invite exceptions.

An example INVEX token:
INVEX

4.11 MAXLIST

MAXLIST=mode:number[,mode:number[,...]]

The MAXLIST parameter limits how many "variable" modes of type A that have been defined in the CHANMODES (Section 4.3) token a client may set in total on a channel. The value MUST be specified and is a set of "mode:number" pairs, where "mode" refers to a type A mode defined in the CHANMODES (Section 4.3) token and "number" indicates how many of the given modes combined a client may issue on a channel.

A client MUST NOT make any assumptions about how many of the given modes may actually exist on the channel. The limit applies only to the client setting new modes of the given types.

Example MAXLIST tokens:
MAXLIST=beI:25
Indicates that a client may set up to a total of 25 of a combination of "+b", "+e" and "+I" modes.
MAXLIST=b:25,eI:50
Indicates that a client may set up to a total of 25 "+b" modes and up to a total of 50 of a combination of "+e" and "+I" modes.

4.12 MODES

MODES=[number]

The MODES parameter limits how many "variable" modes may be set on a channel by a single MODE command from a client. A "variable" mode is defined as being type A, B and C as defined for the CHANMODES (Section 4.3) parameter, and the channel modes specified in the PREFIX (Section 4.15) parameter.

A client SHOULD NOT issue more "variable" modes than this in a single

MODE command. A server MAY however issue more "variable" modes than this value in a single MODE command. The value is OPTIONAL and when is not specified indicates that no limit is placed upon "variable" modes. The value, if specified, MUST be numeric.

An example MODES token:
MODES=3
Limits the number of "variable" modes from a client to the server to 3 per MODE command.

4.13 NETWORK

NETWORK=string

The NETWORK parameter is for informational purposes only and defines the name of the IRC network that the client is connected to. The value MUST be specified. A client SHOULD NOT use the value to make assumptions about supported features on the server.

An example NETWORK token:
NETWORK=EFnet
Indicates the client is connected to the EFnet IRC network.

4.14 NICKLEN

NICKLEN=number

The NICKLEN parameter specifies the maximum length of a nickname that a client can use. A client elsewhere on the network MAY use a nick length of higher value. The value MUST be specified and MUST be numeric.

An example NICKLEN token:
NICKLEN=9
Limits the length of a nickname to 9 characters

4.15 PREFIX

PREFIX=[(modes)prefixes]

Within channels, clients can have various different statuses, denoted by single character "prefixes". The PREFIX parameter specifies these prefixes and the channel mode character that it is mapped to. There is a one-to-one mapping between prefixes and channel modes. The prefixes are in descending order, from the prefix that gives the most privileges to the prefix that gives the least.

The value is OPTIONAL and when it is not specified indicates that no prefixes are supported.

An example PREFIX token:
PREFIX=(ov)@+
Denotes the at character ('@') as mapping to the channel mode denoted by the letter 'o', and the plus character ('+') as mapping to the channel mode denoted by the letter 'v'.

4.16 SAFELIST

SAFELIST

The SAFELIST parameter indicates that the client may request a "LIST" command from the server, without being disconnected by the large volume of data the LIST command generates. The SAFELIST parameter MUST NOT be specified with a value.

An example SAFELIST token:
SAFELIST

4.17 SILENCE

SILENCE=number

The SILENCE parameter indicates the maximum number of entries a user may have in their silence list. The value is OPTIONAL and if it is not specified indicates SILENCE support is not available.

Whilst a formal definition of the SILENCE command is outside the scope of this document, it is generally a list of masks of equivalent form to those defined as type A in the CHANMODES (Section 4.3) parameter. Any messages, as defined in RFC 2812 [3] section 3.3, that originate from another client matching the given mask, with a destination of the client itself will be dropped by the server.

An example SILENCE token:
SILENCE=15
Indicates a client may have upto 15 masks in their silence list.

4.18 STATUSMSG

STATUSMSG=string

The STATUSMSG parameter indicates that the server supports a method for the client to send a message via the NOTICE command to those people on a channel with the specified status.

The value MUST be specified and MUST be a non-delimited list of prefixes that have been defined in the PREFIX (Section 4.15) parameter. The server MUST NOT advertise a character in STATUSMSG which is also present in CHANTYPES (Section 4.5).

An example STATUSMSG token:
STATUSMSG=@+
Presuming the hash character ('#') is defined within the CHANTYPES (Section 4.5) parameter, allows the client to send a NOTICE command to "@#channel" and "+#channel".

4.19 TARGMAX

TARGMAX=[cmd:number,cmd:number,...]

Certain commands from a client MAY contain multiple targets, delimited by a comma character (','). The TARGMAX parameter defines the maximum number of targets allowed for commands which accept multiple targets. The value is OPTIONAL and is a set of "cmd:number" pairs, where "cmd" refers to a command the client MAY send to the server, and "number" refers to the maximum targets for this command. A client MUST treat the "cmd" field as case insensitive.

If the number is not specified for a particular command, then the command does not have a limit on the number of targets. The server MUST specify all commands available to the user which support multiple targets.

If the TARGMAX parameter is not advertised, or a value is not sent then a client SHOULD assume that no commands except the "JOIN" and "PART" commands accept multiple parameters.

An example TARGMAX token:
TARGMAX=PRIVMSG:3,WHOIS:1,JOIN:
Indicates that a client could issue 3 targets to a PRIVMSG command, 1 target to a WHOIS command and an unlimited amount of targets to a JOIN command.

4.20 TOPICLEN

TOPICLEN=number

The TOPICLEN parameter specifies the maximum length of a topic, defined in RFC 1459 [2] section 4.2.4 that a client may set on a channel. A channel on the network MAY have a topic with a longer length. The value MUST be specified and MUST be numeric.

An example TOPICLEN token:
TOPICLEN=120
Limits the length of a topic to 120 characters.

4.21 WATCH

WATCH=number

The WATCH parameter indicates the maximum number of nicknames a user may have in their watch list. The value MUST be specified.

Whilst a formal definition of the WATCH command is outside the scope of this document, it is generally used a method for clients to have the server notify them when a given nickname joins or leaves the network. It is designed to replace repetitive use by clients of the ISON command, as defined in RFC 1459 [2] section 5.8.

An example WATCH token:
WATCH=100
Indicates that a client may have upto 100 nicks in their watch list.

5. Differences to existing implementations

A number of differences exist between this specification and existing implementations in current IRC server software.

5.1 Conflicts with RFC2812

RFC 2812 [3] section 5.1, defines a numeric reply "RPL_BOUNCE" with the associated numeric "005". This conflicts with the numeric defined within this document for RPL_ISUPPORT. As RFC 2812 [3] is an Informational RFC and 005 has been widely adopted as the de-facto standard numeric for RPL_ISUPPORT, this is not seen as problematic. Moreover, the only server implementation known to implement RFC 2812 [3] moved the RPL_BOUNCE numeric to 010 and adopted the RPL_ISUPPORT numeric as 005.

5.2 Traditional EXCEPTS/INVEX usage

The EXCEPTS (Section 4.9) and INVEX (Section 4.10) parameters traditionally take no argument. Whilst they indicate the presence of these features on the server, they rely on these features using the +e and +I channel modes respectively. The argument value described provides extra flexibility whilst retaining backwards compatibility.

5.3 MAXBANS

The MAXBANS parameter was replaced by the MAXLIST (Section 4.11) parameter. MAXBANS was considered non-useful, because of its ambiguous meaning; two of the largest IRC networks for example could not agree whether "MAXBANS=n" was equivalent to "MAXLIST=beI:n" or "MAXLIST=b:n,e:n,I:n". MAXLIST is also considered considerably more flexible and can more accurately define the servers behaviour.

5.4 MAXCHANNELS

The MAXCHANNELS parameter was replaced by the CHANLIMIT (Section 4.2) parameter. Some IRC server implementations did not apply any limits to server-local "&" channels, the MAXCHANNELS parameter did not reflect this and so the CHANLIMIT (Section 4.2) parameter was introduced to replace MAXCHANNELS, as it can more accurately define the server implementation.

5.5 MAXTARGETS

The MAXTARGETS parameter was replaced by the TARGMAX (Section 4.19) parameter. As which commands MAXTARGETS applied to is unclear and different target limits can often apply to different commands, it was believed the TARGMAX parameter could more accurately define which commands may contain multiple targets.

5.6 WALLCHOPS

The WALLCHOPS parameter was replaced by the STATUSMSG (Section 4.18) parameter. It is believed the STATUSMSG (Section 4.18) parameter is more flexible. Some IRC server implementations also implemented a "WALLCHOPS" command, and it was unclear whether the parameter indicated support for this command or not. This behaviour is still unclear.

6. IANA Considerations

This memo does not raise any IANA considerations.

7. Security Considerations

This memo does not raise any security considerations.

8. Acknowledgments

The authors gratefully acknowledges the contributions of Bill Fenner ("fenestro"), Perry Lorier ("Isomer"), Kurt Roeckx ("Q") and John Midgley ("CrazyEddy") in the preparation of this document.

This document is heavily based on a previous document entitled "The 005 numeric".

9 References

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

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

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

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

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

[6] Bradner, S., "IETF Rights in Contributions", BCP 78, RFC 3667, February 2004.

Authors' Addresses

Lee Hardy
ircd-ratbox development team

EMail: lee@leeh.co.uk
URI: http://www.leeh.co.uk

Edward Brocklesby
ircd-ratbox development team
57 Williamson Way
Oxford OX4 4TU
UK

Kevin L. Mitchell
Undernet IRC Network
38 Eighth St., Apt. 7
Cambridge, Massachusetts 02141
US

Phone: +1-617-230-1021
EMail: klmitch@mit.edu
URI: http://www.mit.edu/~klmitch/

Appendix A. Examples

The following examples show various RPL_ISUPPORT messages from various IRC server software.

Example network

```
       :server 005 nickname foo
```

Appendix B. ABNF Description of Capabilities

This section summarizes the ABNF [4] description of the RPL_ISUPPORT extension.

Appendix C. ChangeLog

Note to RFC Editor: This section may be removed on publication as an RFC.

Here is a log of changes to this document:

2005-01-28 LH Initial draft written, borrowing heavily from the old i-d by Edward Brocklesby.

2005-01-29 LH

* Fleshed out information on CPRIVMSG, CNOTICE, WATCH and SILENCE tokens.
* Added CHANNELLEN token.
* Added TOPICLEN token.
* Added TARGMAX token.

2005-02-02 LH

* Documented differences to traditional EXCEPTS/INVEX usage.
* Added the old WALLCHOPS token under differences.
* Added the old MAXBANS token under differences.
* Added the old MAXTARGETS token under differences.
* Added the old MAXCHANNELS token under differences.

2005-02-03 LH

* "Add r remote" to "Add or remove" in CHANMODES.
* Make SILENCE= indicate its unsupported.
* Remove some stuff from STATUSMSG thats beyond the scope of this document.
* Clarify ETRACE modifiers.

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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