Charybdis operator guide Documentation

Release 3.5

Jilles Tjoelker

Mar 25, 2017

Contents

Contents:

CHAPTER 1

Scope of this document

This document describes the commands and functions available to operators in the charybdis ircd, as used on AthemeNet.

This document, and various ideas for features of charybdis, have been taken from dancer-ircd/hyperion, the ircd used on freenode, mainly written by Andrew Suffield and Jilles Tjoelker.

While this document may be of some interest to the users of charybdis servers, it is intended as a reference for network staff.

Charybdis is based on ircd-ratbox 2.1.4, although much has changed. ircd-ratbox is commonly used on efnet, and some other networks.

CHAPTER 2

User modes

+a, server administrator

This vanity usermode is used to denote a server administrator in WHOIS output. All local "admin" privileges are independent of it, though services packages may grant extra privileges to `+a` users.

+D, deaf

Note: This is a user umode, which anybody can set. It is not specific to operators.

Users with the `+D` umode set will not receive messages sent to channels. Joins, parts, topic changes, mode changes, etc are received as normal, as are private messages.

Support of this umode is indicated by the `DEAF` token in `RPL_ISUPPORT` (005); the parameter indicates the letter of the umode. Note that several common IRCD implementations have an umode like this (typically `+d`) but do not have the token in 005.

+g, Caller ID

Note: This is a user umode, which anybody can set. It is not specific to operators.

Users with the `+g` umode set will only receive private messages from users on a session-defined whitelist, defined by the `/accept` command. If a user who is not on the whitelist attempts to send a private message, the target user will receive a rate-limited notice saying that the user wishes to speak to them.

Network operators are not affected by the callerid whitelist system in the event that they need to speak to users who have it enabled.

Support of this umode is indicated by the CALLERID token in RPL_ISUPPORT (005); the optional parameter indicates the letter of the umode, otherwise +g.

+i, invisible

Note: This is a user umode, which anybody can set. It is not specific to operators.

Invisible users do not show up in WHO and NAMES unless you can see them.

+l, receive locops

LOCOPS is a version of OPERWALL that is sent to opers on a single server only. With cluster{} and shared{} blocks they can optionally be propagated further.

Unlike OPERWALL, any oper can send and receive LOCOPS.

+o, operator

This indicates global operator status.

+Q, disable forwarding

Note: This is a user umode, which anybody can set. It is not specific to operators.

This umode prevents you from being affected by channel forwarding. If enabled on a channel, channel forwarding sends you to another channel if you could not join. See channel mode +f for more information.

+R, reject messages from unauthenticated users

Note: This is a user umode, which anybody can set. It is not specific to operators.

If a user has the +R umode set, then any users who are not authenticated will receive an error message if they attempt to send a private message or notice to the +R user.

Opers and accepted users (like in +g) are exempt. Unlike +g, the target user is not notified of failed messages.

+s, receive server notices

This umode allows an oper to receive server notices. The requested types of server notices are specified as a parameter ("snomask") to this umode.

+s, network service

Note: This umode can only be set by servers named in a service{} block.

This umode grants various features useful for services. For example, clients with this umode cannot be kicked or deopped on channels, can send to any channel, do not show channels in WHOIS, can be the target of services aliases and do not appear in /stats p. No server notices are sent for hostname changes by services clients; server notices about kills are sent to snomask +k instead of +s.

The exact effects of this umode are variable; no user or oper on an actual charybdis server can set it.

+w, receive wallops

Note: This is a user umode, which anybody can set. It is not specific to operators.

Users with the +w umode set will receive WALLOPS messages sent by opers. Opers with +w additionally receive WALLOPS sent by servers (e.g. remote CONNECT, remote SQUIT, various severe misconfigurations, many services packages).

+z, receive operwall

OPERWALL differs from WALLOPS in that the ability to receive such messages is restricted. Opers with +z set will receive OPERWALL messages.

+Z, SSL user

This umode is set on clients connected via SSL/TLS. It cannot be set or unset after initial connection.

Snomask usage

Usage is as follows:

```
MODE nick +s +/-flags
```

To set snomasks.

```
MODE nick -s
```

To clear all snomasks.

Umode +s will be set if at least one snomask is set.

Umode +s is oper only by default, but even if you allow nonopers to set it, they will not get any server notices.

Meanings of server notice masks

+b, bot warnings

Opers with the +b snomask set will receive warning messages from the server when potential flooders and spambots are detected.

+c, client connections

Opers who have the +c snomask set will receive server notices when clients attach to the local server.

+C, extended client connection notices

Opers who have the +C snomask set will receive server notices when clients attach to the local server. Unlike the +c snomask, the information is displayed in a format intended to be parsed by scripts, and includes the two unused fields of the USER command.

+d, debug

The +d snomask provides opers extra information which may be of interest to debuggers. It will also cause the user to receive server notices if certain assertions fail inside the server. Its precise meaning is variable. Do not depend on the effects of this snomask as they can and will change without notice in later revisions.

+f, full warning

Opers with the +f snomask set will receive notices when a user connection is denied because a connection limit is exceeded (one of the limits in a class{} block, or the total per-server limit settable with /quote set max).

+F, far client connection notices

Note: This snomask is only available if the sno_farconnect.so extension is loaded.

Opers with +F receive server notices when clients connect or disconnect on other servers. The notices have the same format as those from the +c snomask, except that the class is ? and the source server of the notice is the server the user is/was on.

No notices are generated for netsplits and netjoins. Hence, these notices cannot be used to keep track of all clients on the network.

There is no far equivalent of the +C snomask.

+k, server kill notices

Opers with the +k snomask set will receive server notices when services kill users and when other servers kill and save (forced nick change to UID) users. Kills and saves by this server are on +d or +s.

+n, nick change notices

An oper with +n set will receive a server notice every time a local user changes their nick, giving the old and new nicks. This is mostly useful for bots that track all users on a single server.

+r, notices on name rejections

Opers with this snomask set will receive a server notice when somebody tries to use an invalid username, or if a dumb HTTP proxy tries to connect.

+s, generic server notices

This snomask allows an oper to receive generic server notices. This includes kills from opers (except services).

+u, unauthorized connections

This snomask allows an oper to see when users try to connect who do not have an available auth{} block.

+W, whois notifications

Note: This snomask is only available if the sno_whois.so extension is loaded.

Opers with +W receive notices when a WHOIS is executed on them on their server (showing idle time).

+x, extra routing notices

Opers who have the +x snomask set will get notices about servers connecting and disconnecting on the whole network. This includes all servers connected behind the affected link. This can get rather noisy but is useful for keeping track of all linked servers.

+y, spy

Opers with +y receive notices when users try to join RESV'ed ("juped") channels. Additionally, if certain extension modules are loaded, they will receive notices when special commands are used.

+Z, operspy notices

Opers with +Z receive notices whenever an oper anywhere on the network uses operspy.

This snomask can be configured to be only effective for admins.

CHAPTER 3

Channel modes

+b, channel ban

Bans take one parameter which can take several forms. The most common form is `+b nick!user@host`. The wildcards `*` and `?` are allowed, matching zero-or-more, and exactly-one characters respectively. The masks will be trimmed to fit the maximum allowable length for the relevant element. Bans are also checked against the IP address, even if it resolved or is spoofed. CIDR is supported, like `*!*@10.0.0.0/8`. This is most useful with IPv6. Bans are not checked against the real hostname behind any kind of spoof, except if host mangling is in use (e.g. `extensions/ip_cloaking.so`): if the user's host is mangled, their real hostname is checked additionally, and if a user has no spoof but could enable mangling, the mangled form of their hostname is checked additionally. Hence, it is not possible to evade bans by toggling host mangling.

The second form (extban) is `+b $type` or `+b $type:data`. type is a single character (case insensitive) indicating the type of match, optionally preceded by a tilde (~) to negate the comparison. data depends on type. Each type is loaded as a module. The available types (if any) are listed in the `EXTBAN` token of the 005 (`RPL_ISUPPORT`) numeric. See `doc/extban.txt` in the source distribution for more information.

If no parameter is given, the list of bans is returned. All users can use this form. The plus sign should also be omitted.

Matching users will not be allowed to join the channel or knock on it. If they are already on the channel, they may not send to it or change their nick.

+c, colour filter

This cmode activates the colour filter for the channel. This filters out bold, underline, reverse video, beeps, mIRC colour codes, and ANSI escapes. Note that escape sequences will usually leave cruft sent to the channel, just without the escape characters themselves.

+e, ban exemption

This mode takes one parameter of the same form as bans, which overrides +b and +q bans for all clients it matches.

This can be useful if it is necessary to ban an entire ISP due to persistent abuse, but some users from that ISP should still be allowed in. For example:

```
/mode #channel +be *!*@*.example.com *!*someuser@host3.example.com
```

Only channel operators can see +e changes or request the list.

+f, channel forwarding

This mode takes one parameter, the name of a channel (+f #channel). If the channel also has the +i cmode set, and somebody attempts to join without either being expliticly invited, or having an invex (+I), then they will instead join the channel named in the mode parameter. The client will also be sent a 470 numeric giving the original and target channels.

Users are similarly forwarded if the +j cmode is set and their attempt to join is throttled, if +l is set and there are already too many users in the channel or if +r is set and they are not identified.

Forwards may only be set to +F channels, or to channels the setter has ops in.

Without parameter (/mode #channel f or /mode #channel +f) the forward channel is returned. This form also works off channel.

+F, allow anybody to forward to this

When this mode is set, anybody may set a forward from a channel they have ops in to this channel. Otherwise they have to have ops in this channel.

+g, allow anybody to invite

When this mode is set, anybody may use the INVITE command on the channel in question. When it is unset, only channel operators may use the INVITE command.

When this mode is set together with +i, +j, +l or +r, all channel members can influence who can join.

+i, invite only

When this cmode is set, no client can join the channel unless they have an invex (+I) or are invited with the INVITE command.

+I, invite exception (invex)

This mode takes one parameter of the same form as bans. Matching clients do not need to be invited to join the channel when it is invite-only (+i). Unlike the INVITE command, this does not override +j, +l and +r.

Only channel operators can see +I changes or request the list.

+j, join throttling

This mode takes one parameter of the form n:t, where n and t are positive integers. Only n users may join in each period of t seconds.

Invited users can join regardless of +j, but are counted as normal.

Due to propagation delays between servers, more users may be able to join (by racing for the last slot on each server).

+k, key (channel password)

Taking one parameter, when set, this mode requires a user to supply the key in order to join the channel: `/JOIN #channel key`.

+l, channel member limit

Takes one numeric parameter, the number of users which are allowed to be in the channel before further joins are blocked. Invited users may join regardless.

Due to propagation delays between servers, more users may be able to join (by racing for the last slot on each server).

+L, large ban list

Channels with this mode will be allowed larger banlists (by default, 500 instead of 50 entries for +b, +q, +e and +I together). Only network operators with resv privilege may set this mode.

+m, moderated

When a channel is set +m, only users with +o or +v on the channel can send to it.

Users can still knock on the channel or change their nick.

+n, no external messages

When set, this mode prevents users from sending to the channel without being in it themselves. This is recommended.

+o, channel operator

This mode takes one parameter, a nick, and grants or removes channel operator privilege to that user. Channel operators have full control over the channel, having the ability to set all channel modes except +L and +P, and kick users. Like voiced users, channel operators can always send to the channel, overriding +b, +m and +q modes and the per-channel flood limit. In most clients channel operators are marked with an '@' sign.

The privilege is lost if the user leaves the channel or server in any way.

Most networks will run channel registration services (e.g. ChanServ) which ensure the founder (and users designated by the founder) can always gain channel operator privileges and provide some features to manage the channel.

+p, paranoid channel

When set, the KNOCK command cannot be used on the channel to request an invite, and users will not be shown the channel in WHOIS replies unless they are on it. Unlike in traditional IRC, +p and +s can be set together.

+P, permanent channel

Channels with this mode (which is accessible only to network operators with resv privilege) set will not be destroyed when the last user leaves.

This makes it less likely modes, bans and the topic will be lost and makes it harder to abuse network splits, but also causes more unwanted restoring of old modes, bans and topics after long splits.

+q, quiet

This mode behaves exactly like +b (ban), except that the user may still join the channel. The net effect is that they cannot knock on the channel, send to the channel or change their nick while on channel.

+Q, block forwarded users

Channels with this mode set are not valid targets for forwarding. Any attempt to forward to this channel will be ignored, and the user will be handled as if the attempt was never made (by sending them the relevant error message).

This does not affect the ability to set +f.

+r, block unidentified

When set, this mode prevents unidentified users from joining. Invited users can still join.

+s, secret channel

When set, this mode prevents the channel from appearing in the output of the LIST, WHO and WHOIS command by users who are not on it. Also, the server will refuse to answer WHO, NAMES, TOPIC and LIST queries from users not on the channel.

+t, topic limit

When set, this mode prevents users who are not channel operators from changing the topic.

+v, voice

This mode takes one parameter, a nick, and grants or removes voice privilege to that user. Voiced users can always send to the channel, overriding +b, +m and +q modes and the per-channel flood limit. In most clients voiced users are marked with a plus sign.

The privilege is lost if the user leaves the channel or server in any way.

+z, reduced moderation

When +z is set, the effects of +m, +b and +q are relaxed. For each message, if that message would normally be blocked by one of these modes, it is instead sent to all channel operators. This is intended for use in moderated debates.

Note that +n is unaffected by this. To silence a given user completely, remove them from the channel.

CHAPTER 4

User Commands

Standard IRC commands are not listed here. Several of the commands in the operator commands chapter can also be used by normal users.

ACCEPT

```
ACCEPT nick, -nick, ...
```

Adds or removes users from your accept list for umode +g and +R. Users are automatically removed when they quit, split or change nick.

```
ACCEPT *
```

Lists all users on your accept list.

Support of this command is indicated by the `CALLERID` token in `RPL_ISUPPORT` (005); the optional parameter indicates the letter of the "only allow accept users to send private messages" umode, otherwise +g. In charybdis this is always +g.

CNOTICE

```
CNOTICE nick channel :text
```

Providing you are opped (+o) or voiced (+v) in channel, and nick is a member of channel, `CNOTICE` generates a `NOTICE` towards nick.

`CNOTICE` bypasses any anti-spam measures in place. If you get "Targets changing too fast, message dropped", you should probably use this command, for example sending a notice to every user joining a certain channel.

As of charybdis 3.1, `NOTICE` automatically behaves as `CNOTICE` if you are in a channel fulfilling the conditions.

Support of this command is indicated by the `CNOTICE` token in `RPL_ISUPPORT` (005).

CPRIVMSG

```
CPRIVMSG nick channel :text
```

Providing you are opped (+o) or voiced (+v) in channel, and nick is a member of channel, CPRIVMSG generates a PRIVMSG towards nick.

CPRIVMSG bypasses any anti-spam measures in place. If you get "Targets changing too fast, message dropped", you should probably use this command.

As of charybdis 3.1, PRIVMSG automatically behaves as CPRIVMSG if you are in a channel fulfilling the conditions.

Support of this command is indicated by the CPRIVMSG token in RPL_ISUPPORT (005).

FINDFORWARDS

```
FINDFORWARDS channel
```

Note: This command is only available if the m_findforwards.so extension is loaded.

Displays which channels forward to the given channel (via cmode +f). If there are very many channels the list will be truncated.

You must be a channel operator on the channel or an IRC operator to use this command.

HELP

```
HELP [topic]
```

Displays help information. topic can be INDEX, CREDITS, UMODE, CMODE, SNOMASK or a command name.

There are separate help files for users and opers. Opers can use UHELP to query the user help files.

IDENTIFY

```
IDENTIFY parameters...
```

Note: This command is only available if the m_identify.so extension is loaded.

Sends an identify command to either NickServ or ChanServ. If the first parameter starts with #, the command is sent to ChanServ, otherwise to NickServ. The word IDENTIFY, a space and all parameters are concatenated and sent as a PRIVMSG to the service. If the service is not online or does not have umode +S set, no message will be sent.

The exact syntax for this command depends on the services package in use.

KNOCK

```
KNOCK channel
```

Requests an invite to the given channel. The channel must be locked somehow (+ikl), must not be +p and you may not be banned or quieted. Also, this command is rate limited.

If successful, all channel operators will receive a 710 numeric. The recipient field of this numeric is the channel.

Support of this command is indicated by the `KNOCK` token in `RPL_ISUPPORT` (005).

MONITOR

Server side notify list. This list contains nicks. When a user connects, quits with a listed nick or changes to or from a listed nick, you will receive a 730 numeric if the nick went online and a 731 numeric if the nick went offline.

Support of this command is indicated by the `MONITOR` token in `RPL_ISUPPORT` (005); the parameter indicates the maximum number of nicknames you may have in your monitor list.

You may only use this command once per second.

More details can be found in `doc/monitor.txt` in the source distribution.

```
MONITOR + nick, ...
```

Adds nicks to your monitor list. You will receive 730 and 731 numerics for the nicks.

```
MONITOR - nick, ...
```

Removes nicks from your monitor list. No output is generated for this command.

```
MONITOR C
```

Clears your monitor list. No output is generated for this command.

```
MONITOR L
```

Lists all nicks on your monitor list, using 732 numerics and ending with a 733 numeric.

```
MONITOR S
```

Shows status for all nicks on your monitor list, using 730 and 731 numerics.

CHAPTER 5

Operator Commands

Network management commands

Note: All commands and names are case insensitive. Parameters consisting of one or more separate letters, such as in `MODE`, `STATS` and `WHO`, are case sensitive.

CONNECT

```
CONNECT target [port] [source]
```

Initiate a connection attempt to server target. If a port is given, connect to that port on the target, otherwise use the one given in `ircd.conf`. If source is given, tell that server to initiate the connection attempt, otherwise it will be made from the server you are attached to.

To use the default port with source, specify 0 for port.

SQUIT

```
SQUIT server [reason]
```

Closes down the link to server from this side of the network. If a reason is given, it will be sent out in the server notices on both sides of the link.

REHASH

```
REHASH [BANS | DNS | MOTD | OMOTD | TKLINES | TDLINES | TXLINES | TRESVS |␣
↪REJECTCACHE | HELP] [server]
```

With no parameter given, `ircd.conf` will be reread and parsed. The server argument is a wildcard match of server names.

BANS Rereads `kline.conf`, `dline.conf`, `xline.conf`, `resv.conf` and their .perm variants

DNS Reread `/etc/resolv.conf`.

MOTD Reload the `MOTD` file

OMOTD Reload the operator `MOTD` file

TKLINES Clears temporary `K:lines`.

TDLINES Clears temporary `D:lines`.

TXLINES Clears temporary `X:lines`.

TRESVS Clears temporary reservations.

REJECTCACHE Clears the client rejection cache.

HELP Refreshes the help system cache.

RESTART

```
RESTART server
```

Cause an immediate total shutdown of the IRC server, and restart from scratch as if it had just been executed.

This reexecutes the ircd using the compiled-in path, visible as `SPATH` in `INFO`.

Note: This command cannot be used remotely. The server name is used only as a safety measure.

DIE

```
DIE server
```

Immediately terminate the IRC server, after sending notices to all connected clients and servers

Note: This command cannot be used remotely. The server name is used only as a safety measure.

SET

```
SET [ ADMINSTRING | AUTOCONN | AUTOCONNALL | FLOODCOUNT | IDENTTIMEOUT | MAX |␣
↪OPERSTRING | SPAMNUM | SPAMTIME | SPLITMODE | SPLITNUM | SPLITUSERS ] value
```

The `SET` command sets a runtime-configurable value.

Most of the `ircd.conf` equivalents have a `default_prefix` and are only read on startup. `SET` is the only way to change these at run time.

Most of the values can be queried by omitting value.

ADMINSTRING Sets string shown in `WHOIS` for admins. (umodes +o and +a set, umode +S not set).

AUTOCONN Sets auto-connect on or off for a particular server. Takes two parameters, server name and new state.

To see these values, use `/stats c`. Changes to this are lost on a rehash.

AUTOCONNALL Globally sets auto-connect on or off. If disabled, no automatic connections are done; if enabled, automatic connections are done following the rules for them.

FLOODCOUNT The number of lines allowed to be sent to a connection before throttling it due to flooding. Note that this variable is used for both channels and clients.

For channels, op or voice overrides this; for users, IRC operator status or op or voice on a common channel overrides this.

IDENTTIMEOUT Timeout for requesting ident from a client.

MAX Sets the maximum number of connections to value.

This number cannot exceed maxconnections - `MAX_BUFFER`. maxconnections is the rlimit for number of open files. `MAX_BUFFER` is defined in config.h, normally 60.

`MAXCLIENTS` is an alias for this.

OPERSTRING Sets string shown in `WHOIS` for opers (umode `+o` set, umodes `+a` and `+S` not set).

SPAMNUM Sets how many join/parts to channels constitutes a possible spambot.

SPAMTIME Below this time on a channel counts as a join/part as above.

SPLITMODE Sets splitmode to value:

ON splitmode is permanently on

OFF splitmode is permanently off (default if `no_create_on_split` and `no_join_on_split` are disabled)

AUTO ircd chooses splitmode based on `SPLITUSERS` and `SPLITNUM` (default if `no_create_on_split` or `no_join_on_split` are enabled)

SPLITUSERS Sets the minimum amount of users needed to deactivate automatic splitmode.

SPLITNUM Sets the minimum amount of servers needed to deactivate automatic splitmode. Only servers that have finished bursting count for this.

User management commands

KILL

```
KILL nick [reason]
```

Disconnects the user with the given nick from the server they are connected to, with the reason given, if present, and broadcast a server notice announcing this.

Your nick and the reason will appear on channels.

CLOSE

Closes all connections from and to clients and servers who have not completed registering.

KLINE

```
KLINE [length] [user@host | user@a.b.c.d] [ON servername] [:reason]
```

Adds a `K:line` to `kline.conf` to ban the given `user@host` from using that server.

If the optional parameter length is given, the `K:line` will be temporary (i.e. it will not be stored on disk) and last that long in minutes.

If an IP address is given, the ban will be against all hosts matching that IP regardless of DNS. The IP address can be given as a full address (`192.168.0.1`), as a CIDR mask (`192.168.0.0/24`), or as a glob (`192.168.0.*`).

All clients matching the `K:line` will be disconnected from the server immediately.

If a reason is specified, it will be sent to the client when they are disconnected, and whenever a connection is attempted which is banned.

If the ON part is specified, the `K:line` is set on servers matching the given mask (provided a matching `shared{}` block exists there). Otherwise, if specified in a `cluster{}` block, the `K:Line` will be propagated across the network accordingly.

UNKLINE

```
UNKLINE user@host [ON servername]
```

Will attempt to remove a `K:line` matching `user@host` from `kline.conf`, and will flush a temporary `K:line`.

XLINE

```
XLINE [length] mask [ON servername] [:reason]
```

Works similarly to `KLINE`, but matches against the real name field. The wildcards are `*` (any sequence), `?` (any character), `#` (a digit) and `@` (a letter); wildcard characters can be escaped with a backslash. The sequence `\s` matches a space.

All clients matching the `X:line` will be disconnected from the server immediately.

The reason is never sent to users. Instead, they will be exited with "Bad user info".

If the ON part is specified, the `X:line` is set on servers matching the given mask (provided a matching `shared{}` block exists there). Otherwise, if specified in a `cluster{}` block, the `X:line` will be propagated across the network accordingly.

UNXLINE

```
UNXLINE mask [ON servername]
```

Will attempt to remove an `X:line` from `xline.conf`, and will flush a temporary `X:line`.

RESV

```
RESV [length] [channel | mask] [ON servername] [:reason]
```

If used on a channel, "jupes" the channel locally. Joins to the channel will be disallowed and generate a server notice on `+y`, and users will not be able to send to the channel. Channel jupes cannot contain wildcards.

If used on a nickname mask, prevents local users from using a nick matching the mask (the same wildcard characters as xlines). There is no way to exempt the initial nick from this.

In neither case will current users of the nick or channel be kicked or disconnected.

This facility is not designed to make certain nicks or channels oper-only.

The reason is never sent to users.

If the ON part is specified, the resv is set on servers matching the given mask (provided a matching `shared{}` block exists there). Otherwise, if specified in a `cluster{}` block, the resv will be propagated across the network accordingly.

UNRESV

```
UNRESV [channel | mask] [ON servername]
```

Will attempt to remove a resv from `resv.conf`, and will flush a temporary resv.

DLINE

```
DLINE [length] a.b.c.d [ON servername] [:reason]
```

Adds a `D:line` to `dline.conf`, which will deny any connections from the given IP address. The IP address can be given as a full address (`192.168.0.1`) or as a CIDR mask (`192.168.0.0/24`).

If the optional parameter length is given, the `D:line` will be temporary (i.e. it will not be stored on disk) and last that long in minutes.

All clients matching the `D:line` will be disconnected from the server immediately.

If a reason is specified, it will be sent to the client when they are disconnected, and, if `dline_reason` is enabled, whenever a connection is attempted which is banned.

`D:lines` are less load on a server, and may be more appropriate if somebody is flooding connections.

If the ON part is specified, the `D:line` is set on servers matching the given mask (provided a matching `shared{}` block exists there, which is not the case by default). Otherwise, the D:Line will be set on the local server only.

Only `exempt{}` blocks exempt from `D:lines`. Being a server or having `kline_exempt` in `auth{}` does *not* exempt (different from `K/G/X:lines`).

UNDLINE

```
UNDLINE a.b.c.d [ON servername]
```

Will attempt to remove a `D:line` from `dline.conf`, and will flush a temporary `D:line`.

TESTGECOS

```
TESTGECOS gecos
```

Looks up X:Lines matching the given gecos.

TESTLINE

```
TESTLINE [nick!] [user@host | a.b.c.d]
```

Looks up the given hostmask or IP address and reports back on any `auth{}` blocks, D: or K: lines found. If nick is given, also searches for nick resvs.

For temporary items the number of minutes until the item expires is shown (as opposed to the hit count in STATS q/Q/x/X).

This command will not perform DNS lookups; for best results you must testline a host and its IP form.

The given username should begin with a tilde (~) if identd is not in use. As of charybdis 2.1.1, `no_tilde` and username truncation will be taken into account like in the normal client access check.

As of charybdis 2.2.0, a channel name can be specified and the RESV will be returned, if there is one.

TESTMASK

```
TESTMASK hostmask [gecos]
```

Searches the network for users that match the hostmask and gecos given, returning the number of matching users on this server and other servers.

The hostmask is of the form user@host or user@ip/cidr with * and ? wildcards, optionally preceded by nick!.

The gecos field accepts the same wildcards as xlines.

The IP address checked against is `255.255.255.255` if the IP address is unknown (remote client on a TS5 server) or 0 if the IP address is hidden (`auth{}` spoof).

LUSERS

```
LUSERS [mask] [nick | server]
```

Shows various user and channel counts.

The mask parameter is obsolete but must be used when querying a remote server.

TRACE

```
TRACE [server | nick] [location]
```

With no argument or one argument which is the current server, TRACE gives a list of all connections to the current server and a summary of connection classes.

With one argument which is another server, TRACE displays the path to the specified server, and all servers, opers and -i users on that server, along with a summary of connection classes.

With one argument which is a client, TRACE displays the path to that client, and that client's information.

If location is given, the command is executed on that server; no path is displayed.

When listing connections, type, name and class is shown in addition to information depending on the type:

Try. A server we are trying to make a TCP connection to.

H.S. A server we have established a TCP connection to, but is not yet registered.

???? An incoming connection that has not yet registered as a user or a server ("unknown"). Shows the username, hostname, IP address and the time the connection has been open. It is possible that the ident or DNS lookups have not completed yet, and in any case no tildes are shown here. Unknown connections may not have a name yet.

User A registered unopered user. Shows the username, hostname, IP address, the time the client has not sent anything (as in STATS l) and the time the user has been idle (from PRIVMSG only, as in WHOIS).

Oper Like User, but opered.

Serv A registered server. Shows the number of servers and users reached via this link, who made this connection and the time the server has not sent anything.

ETRACE

```
ETRACE [nick]
```

Shows client information about the given target, or about all local clients if no target is specified.

PRIVS

```
PRIVS [nick]
```

Displays effective operator privileges for the specified nick, or for yourself if no nick is given. This includes all privileges from the operator block, the name of the operator block and those privileges from the auth block that have an effect after the initial connection.

The exact output depends on the server the nick is on, see the matching version of this document. If the remote server does not support this extension, you will not receive a reply.

MASKTRACE

```
MASKTRACE hostmask [gecos]
```

Searches the local server or network for users that match the hostmask and gecos given. Network searches require the `oper_spy` privilege and an '!' before the hostmask. The matching works the same way as TESTMASK.

The hostmask is of the form `user@host` or `user@ip/cidr` with `*` and `?` wildcards, optionally preceded by `nick!`.

The gecos field accepts the same wildcards as xlines.

The IP address field contains `255.255.255.255` if the IP address is unknown (remote client on a TS5 server) or `0` if the IP address is hidden (`auth{}` spoof).

CHANTRACE

```
CHANTRACE channel
```

Displays information about users in a channel. Opers with the `oper_spy` privilege can get the information without being on the channel, by prefixing the channel name with an `!`.

The IP address field contains `255.255.255.255` if the IP address is unknown (remote client on a TS5 server) or `0` if the IP address is hidden (`auth{}` spoof).

SCAN

```
SCAN UMODES +modes-modes [no-list] [list] [global] [list-max number] [mask nick!
↪user@host]
```

Searches the local server or network for users that have the umodes given with + and do not have the umodes given with -. no-list disables the listing of matching users and only shows the count. list enables the listing (default). global extends the search to the entire network instead of local users only. list-max limits the listing of matching users to the given amount. mask causes only users matching the given nick!user@host mask to be selected. Only the displayed host is considered, not the IP address or real host behind dynamic spoofs.

The IP address field contains `255.255.255.255` if the IP address is unknown (remote client on a TS5 server) or 0 if the IP address is hidden (`auth{}` spoof).

Network searches where a listing is given are operspy commands.

CHGHOST

```
CHGHOST nick value
```

Set the hostname associated with a particular nick for the duration of this session. This command is disabled by default because of the abuse potential and little practical use.

Miscellaneous commands

ADMIN

```
ADMIN [nick | server]
```

Shows the information in the `admin{}` block.

INFO

```
INFO [nick | server]
```

Shows information about the authors of the IRC server, and some information about this server instance. Opers also get a list of configuration options.

TIME

```
TIME [nick | server]
```

Shows the local time on the given server, in a human-readable format.

VERSION

```
VERSION [nick | server]
```

Shows version information, a few compile/config options, the SID and the 005 numerics. The 005 numeric will be remapped to 105 for remote requests.

STATS

```
STATS [type] [nick | server]
```

Display various statistics and configuration information.

A Show DNS servers

b Show active nick delays

B Show hash statistics

c Show connect blocks

d Show temporary `D:lines`

D Show permanent `D:lines`

e Show exempt blocks (exceptions to `D:lines`)

E Show events

f Show file descriptors

h Show `hub_mask`/`leaf_mask`

i Show auth blocks, or matched auth blocks

k Show temporary `K:lines`, or matched `K:lines`

K Show permanent `K:lines`, or matched `K:lines`

l Show hostname and link information about the given nick. With a server name, show information about opers and servers on that server; opers get information about all local connections if they query their own server. No hostname is shown for server connections.

L Like l, but show IP address instead of hostname

m Show commands and their usage statistics (total counts, total bytes, counts from server connections)

n Show blacklist blocks (DNS blacklists) with hit counts since last rehash and (parenthesized) reference counts. The reference count shows how many clients are waiting on a lookup of this blacklist or have been found and are waiting on registration to complete.

o Show operator blocks

O Show privset blocks

p Show logged on network operators which are not set AWAY.

P Show listen blocks (ports)

q Show temporarily resv'ed nicks and channels with hit counts

Q Show permanently resv'ed nicks and channels with hit counts since last rehash bans

r Show resource usage by the ircd

t Show generic server statistics about local connections

u Show server uptime

U Show shared (c), cluster (C) and service (s) blocks

v Show connected servers and brief status

x Show temporary `X:lines` with hit counts

X Show permanent `X:lines` with hit counts since last rehash bans

y Show class blocks

z Show memory usage statistics

Z Show ziplinks statistics

? Show connected servers and link information about them

WALLOPS

```
WALLOPS :message
```

Sends a WALLOPS message to all users who have the +w umode set. This is for things you don't mind the whole network knowing about.

OPERWALL

```
OPERWALL :message
```

Sends an OPERWALL message to all opers who have the +z umode set. +z is restricted, OPERWALL should be considered private communications.

CHAPTER 6

Oper privileges

These are specified in privset{}.

oper:admin, server administrator

Various privileges intended for server administrators. Among other things, this automatically sets umode +a and allows loading modules.

oper:die, die and restart

This grants permission to use `DIE` and `RESTART`, shutting down or restarting the server.

oper:global_kill, global kill

Allows using `KILL` on users on any server.

oper:hidden, hide from /stats p

This privilege currently does nothing, but was designed to hide bots from /stats p so users will not message them for help.

oper:hidden_admin, hidden administrator

This grants everything granted to the oper:admin privilege, except the ability to set umode +a. If both oper:admin and oper:hidden_admin are possessed, umode +a can still not be used.

oper:kline, kline and dline

Allows using KLINE and DLINE, to ban users by user@host mask or IP address.

oper:local_kill, kill local users

This grants permission to use KILL on users on the same server, disconnecting them from the network.

oper:mass_notice, global notices and wallops

Allows using server name ($$mask) and hostname ($#mask) masks in NOTICE and PRIVMSG to send a message to all matching users, and allows using the WALLOPS command to send a message to all users with umode +w set.

oper:operwall, send/receive operwall

Allows using the OPERWALL command and umode +z to send and receive operwalls.

oper:rehash, rehash

Allows using the REHASH command, to rehash various configuration files or clear certain lists.

oper:remoteban, set remote bans

This grants the ability to use the ON argument on DLINE/KLINE/XLINE/RESV and UNDLINE/UNKLINE/UNXLINE/UNRESV to set and unset bans on other servers, and the server argument on REHASH. This is only allowed if the oper may perform the action locally, and if the remote server has a shared{} block.

Note: If a cluster{} block is present, bans are sent remotely even if the oper does not have oper:remoteban privilege.

oper:resv, channel control

This allows using /resv, /unresv and changing the channel modes +L and +P.

oper:routing, remote routing

This allows using the third argument of the CONNECT command, to instruct another server to connect somewhere, and using SQUIT with an argument that is not locally connected. (In both cases all opers with +w set will be notified.)

oper:spy, use operspy

This allows using `/mode !#channel`, `/whois !nick`, `/who !#channel`, `/chantrace !#channel`, `/topic !#channel`, `/who !mask`, `/masktrace !user@host :gecos` and `/scan umodes +modes-modes global list` to see through secret channels, invisible users, etc.

All operspy usage is broadcasted to opers with snomask `+Z` set (on the entire network) and optionally logged. If you grant this to anyone, it is a good idea to establish concrete policies describing what it is to be used for, and what not.

If `operspy_dont_care_user_info` is enabled, `/who mask` is operspy also, and `/who !mask`, `/who mask`, `/masktrace !user@host :gecos` and `/scan umodes +modes-modes global list` do not generate `+Z` notices or logs.

oper:unkline, unkline and undline

Allows using `UNKLINE` and `UNDLINE`.

oper:xline, xline and unxline

Allows using `XLINE` and `UNXLINE`, to ban/unban users by realname.

snomask:nick_changes, see nick changes

Allows using snomask `+n` to see local client nick changes. This is designed for monitor bots.

CHAPTER 7

Server config file format

General format

The config file consists of a series of BIND-style blocks. Each block consists of a series of values inside it which pertain to configuration settings that apply to the given block.

Several values take lists of values and have defaults preset inside them. Prefix a keyword with a tilde (~) to override the default and disable it.

A line may also be a .include directive, which is of the form:

```
.include "file"
```

and causes file to be read in at that point, before the rest of the current file is processed. Relative paths are first tried relative to `PREFIX` and then relative to `ETCPATH` (normally `PREFIX`/etc).

Anything from a # to the end of a line is a comment. Blank lines are ignored. C-style comments are also supported.

Specific blocks and directives

Not all configuration blocks and directives are listed here, only the most common ones. More blocks and directives will be documented in later revisions of this manual.

loadmodule directive

```
loadmodule "text";
```

Loads a module into the IRCd. In charybdis 1.1, most modules are automatically loaded in. In future versions, it is intended to remove this behaviour as to allow for easy customization of the IRCd's featureset.

serverinfo {} block

```
serverinfo {
     name = "text";
     sid = "text";
     description = "text";
     network_name = "text";
     network_desc = "text";
     hub = boolean;
     vhost = "text";
     vhost6 = "text";
};
```

The serverinfo { } block defines the core operational parameters of the IRC server.

serverinfo {} variables

name The name of the IRC server that you are configuring. This must contain at least one dot. It is not necessarily equal to any DNS name. This must be unique on the IRC network.

sid A unique ID which describes the server. This consists of one digit and two characters which can be digits or letters.

description A user-defined field of text which describes the IRC server. This information is used in `/links` and `/whois` requests. Geographical location information could be a useful use of this field, but most administrators put a witty saying inside it instead.

network_name The name of the IRC network that this server will be a member of. This is used in the welcome message and `NETWORK=` in 005.

hub A boolean which defines whether or not this IRC server will be serving as a hub, i.e. have multiple servers connected to it.

vhost An optional text field which defines an IP from which to connect outward to other IRC servers.

vhost6 An optional text field which defines an IPv6 IP from which to connect outward to other IRC servers.

admin {} block

```
admin {
    name = "text";
        description = "text";
        email = "text";
};
```

This block provides the information which is returned by the `ADMIN` command.

name The name of the administrator running this service.

description The description of the administrator's position in the network.

email A point of contact for the administrator, usually an e-mail address.

class {} block

```
class "name" {
        ping_time = duration;
        number_per_ident = number;
        number_per_ip = number;
        number_per_ip_global = number;
        cidr_ipv4_bitlen = number;
        cidr_ipv6_bitlen = number;
        number_per_cidr = number;
        max_number = number;
        sendq = size;
};

class "name" {
        ping_time = duration;
        connectfreq = duration;
        max_number = number;
        sendq = size;
};
```

Class blocks define classes of connections for later use. The class name is used to connect them to other blocks in the config file (auth{} and connect{}). They must be defined before they are used.

Classes are used both for client and server connections, but most variables are different.

class {} variables: client classes

ping_time The amount of time between checking pings for clients, e.g.: 2 minutes

number_per_ident The amount of clients which may be connected from a single identd username on a per-IP basis, globally. Unidented clients all count as the same username.

number_per_ip The amount of clients which may be connected from a single IP address.

number_per_ip_global The amount of clients which may be connected globally from a single IP address.

cidr_ipv4_bitlen The netblock length to use with CIDR-based client limiting for IPv4 users in this class (between 0 and 32).

cidr_ipv6_bitlen The netblock length to use with CIDR-based client limiting for IPv6 users in this class (between 0 and 128).

number_per_cidr The amount of clients which may be connected from a single netblock.

If this needs to differ between IPv4 and IPv6, make different classes for IPv4 and IPv6 users.

max_number The maximum amount of clients which may use this class at any given time.

sendq The maximum size of the queue of data to be sent to a client before it is dropped.

class {} variables: server classes

ping_time The amount of time between checking pings for servers, e.g.: 2 minutes

connectfreq The amount of time between autoconnects. This must at least be one minute, as autoconnects are evaluated with that granularity.

max_number The amount of servers to autoconnect to in this class. More precisely, no autoconnects are done if the number of servers in this class is greater than or equal max_number

sendq The maximum size of the queue of data to be sent to a server before it is dropped.

auth {} block

```
auth {
    user = "hostmask";
    password = "text";
    spoof = "text";
    flags = list;
    class = "text";
};
```

auth {} blocks allow client connections to the server, and set various properties concerning those connections.

Auth blocks are evaluated from top to bottom in priority, so put special blocks first.

auth {} variables

user A hostmask (`user@host`) that the auth {} block applies to. It is matched against the hostname and IP address (using :: shortening for IPv6 and prepending a 0 if it starts with a colon) and can also use CIDR masks. You can have multiple user entries.

password An optional password to use for authenticating into this auth{} block. If the password is wrong the user will not be able to connect (will not fall back on another auth{} block).

spoof An optional fake hostname (or `user@host`) to apply to users authenticated to this auth{} block. In `STATS i` and `TESTLINE`, an equals sign (=) appears before the `user@host` and the spoof is shown.

flags A list of flags to apply to this `auth{}` block. They are listed below. Some of the flags appear as a special character, parenthesized in the list, before the `user@host` in `STATS i` and `TESTLINE`.

class A name of a class to put users matching this auth{} block into.

auth {} flags

encrypted The password used has been encrypted.

spoof_notice Causes the IRCd to send out a server notice when activating a spoof provided by this auth{} block.

exceed_limit (>) Users in this auth{} block can exceed class-wide limitations.

dnsbl_exempt ($) Users in this auth{} block are exempted from DNS blacklist checks. However, they will still be warned if they are listed.

kline_exempt (^) Users in this auth{} block are exempted from DNS blacklists, k:lines and x:lines.

spambot_exempt Users in this auth{} block are exempted from spambot checks.

shide_exempt Users in this auth{} block are exempted from some serverhiding effects.

jupe_exempt Users in this auth{} block do not trigger an alarm when joining juped channels.

resv_exempt Users in this auth{} block may use reserved nicknames and channels.

> **Note:** The initial nickname may still not be reserved.

flood_exempt (|) Users in this auth{} block may send arbitrary amounts of commands per time unit to the server. This does not exempt them from any other flood limits. You should use this setting with caution.

no_tilde (-) Users in this auth{} block will not have a tilde added to their username if they do not run identd.

need_ident (+) Users in this auth{} block must have identd, otherwise they will be rejected.

need_ssl Users in this auth{} block must be connected via SSL/TLS, otherwise they will be rejected.

need_sasl Users in this auth{} block must identify via SASL, otherwise they will be rejected.

exempt {} block

```
exempt {
    ip = "ip";
};
```

An exempt block specifies IP addresses which are exempt from `D:lines` and throttling. Multiple addresses can be specified in one block. Clients coming from these addresses can still be `K/G/X:lined` or banned by a DNS blacklist unless they also have appropriate flags in their auth{} block.

exempt {} variables

ip The IP address or CIDR range to exempt.

privset {} block

```
privset {
    extends = "name";
    privs = list;
};
```

A privset (privilege set) block specifies a set of operator privileges.

privset {} variables

extends An optional privset to inherit. The new privset will have all privileges that the given privset has.

privs Privileges to grant to this privset. These are described in the operator privileges section.

operator {} block

```
operator "name" {
     user = "hostmask";
     password = "text";
     rsa_public_key_file = "text";
     umodes = list;
     snomask = "text";
     flags = list;
};
```

Operator blocks define who may use the OPER command to gain extended privileges.

operator {} variables

user A hostmask that users trying to use this operator { } block must match. This is checked against the original host and IP address; CIDR is also supported. So auth { } spoofs work in operator { } blocks; the real host behind them is not checked. Other kind of spoofs do not work in operator { } blocks; the real host behind them is checked.

Note that this is different from charybdis 1.x where all kinds of spoofs worked in operator { } blocks.

password A password used with the OPER command to use this operator { } block. Passwords are encrypted by default, but may be unencrypted if ~encrypted is present in the flags list.

rsa_public_key_file An optional path to a RSA public key file associated with the operator { } block. This information is used by the CHALLENGE command, which is an alternative authentication scheme to the traditional OPER command.

umodes A list of usermodes to apply to successfully opered clients.

snomask An snomask to apply to successfully opered clients.

privset The privilege set granted to successfully opered clients. This must be defined before this operator{ } block.

flags A list of flags to apply to this operator{ } block. They are listed below.

operator {} flags

encrypted The password used has been encrypted. This is enabled by default, use ~encrypted to disable it.

need_ssl Restricts use of this operator{ } block to SSL/TLS connections only.

connect {} block

```
connect "name" {
    host = "text";
    send_password = "text";
    accept_password = "text";
    port = number;
    hub_mask = "mask";
    leaf_mask = "mask";
    class = "text";
    flags = list;
    aftype = protocol;
};
```

Connect blocks define what servers may connect or be connected to.

connect {} variables

host The hostname or IP to connect to.

> **Note:**
>
> **Furthermore, if a hostname is used, it must have an** `A` or `AAAA` record (no `CNAME`) and it must be the primary hostname for inbound connections to work.
>
> IPv6 addresses must be in `::` shortened form; addresses which then start with a colon must be prepended with a zero, for example `0::1`.

send_password The password to send to the other server.

accept_password The password that should be accepted from the other server.

port The port on the other server to connect to.

hub_mask An optional domain mask of servers allowed to be introduced by this link. Usually, "*" is fine. Multiple hub_masks may be specified, and any of them may be introduced. Violation of hub_mask and leaf_mask restrictions will cause the local link to be closed.

leaf_mask An optional domain mask of servers not allowed to be introduced by this link. Multiple leaf_masks may be specified, and none of them may be introduced. leaf_mask has priority over hub_mask.

class The name of the class this server should be placed into.

flags A list of flags concerning the connect block. They are listed below.

aftype The protocol that should be used to connect with, either ipv4 or ipv6. This defaults to ipv4 unless host is a numeric IPv6 address.

connect {} flags

encrypted The value for accept_password has been encrypted.

autoconn The server should automatically try to connect to the server defined in this connect {} block if it's not connected already and max_number in the class is not reached yet.

compressed Ziplinks should be used with this server connection. This compresses traffic using zlib, saving some bandwidth and speeding up netbursts.

If you have trouble setting up a link, you should turn this off as it often hides error messages.

topicburst Topics should be bursted to this server.

This is enabled by default.

listen {} block

```
listen {
     host = "text";
     port = number;
};
```

A listen block specifies what ports a server should listen on.

listen {} variables

host An optional host to bind to. Otherwise, the ircd will listen on all available hosts.

port A port to listen on. You can specify multiple ports via commas, and define a range by seperating the start and end ports with two dots (..).

modules {} block

```
modules {
    path = "text";
    module = text;
};
```

The modules block specifies information for loadable modules.

modules {} variables

path Specifies a path to search for loadable modules.

module Specifies a module to load, similar to loadmodule.

general {} block

```
modules {
    values
};
```

The general block specifies a variety of options, many of which were in `config.h` in older daemons. The options are documented in `reference.conf`.

channel {} block

```
modules {
    values
};
```

The channel block specifies a variety of channel-related options, many of which were in `config.h` in older daemons. The options are documented in `reference.conf`.

serverhide {} block

```
modules {
    values
};
```

The serverhide block specifies options related to server hiding. The options are documented in `reference.conf`.

blacklist {} block

```
blacklist {
    host = "text";
    reject_reason = "text";
};
```

The blacklist block specifies DNS blacklists to check. Listed clients will not be allowed to connect. IPv6 clients are not checked against these.

Multiple blacklists can be specified, in pairs with first host then reject_reason.

blacklist {} variables

host The DNSBL to use.

reject_reason The reason to send to listed clients when disconnecting them.

alias {} block

```
alias "name" {
    target = "text";
};
```

Alias blocks allow the definition of custom commands. These commands send `PRIVMSG` to the given target. A real command takes precedence above an alias.

alias {} variables

target The target nick (must be a network service (umode `+S`)) or user@server. In the latter case, the server cannot be this server, only opers can use user starting with "opers" reliably and the user is interpreted on the target server only so you may need to use nick@server instead).

cluster {} block

```
cluster {
    name = "text";
    flags = list;
};
```

The cluster block specifies servers we propagate things to automatically. This does not allow them to set bans, you need a separate shared{} block for that.

Having overlapping cluster{} items will cause the command to be executed twice on the target servers. This is particularly undesirable for ban removals.

The letters in parentheses denote the flags in `/stats U`.

cluster {} variables

name The server name to share with, this may contain wildcards and may be stacked.

flags The list of what to share, all the name lines above this (up to another flags entry) will receive these flags. They are listed below.

cluster {} flags

kline (K) Permanent `K:lines`

tkline (k) Temporary `K:lines`

unkline (U) `K:line` removals

xline (X) Permanent `X:lines`

txline (x) Temporary `X:lines`

unxline (Y) `X:line` removals

resv (Q) Permanently reserved nicks/channels

tresv (q) Temporarily reserved nicks/channels

unresv (R) `RESV` removals

locops (L) `LOCOPS` messages (sharing this with * makes `LOCOPS` rather similar to `OPERWALL` which is not useful)

all All of the above

shared {} block

```
shared {
    oper = "user@host", "server";
    flags = list;
};
```

The shared block specifies opers allowed to perform certain actions on our server remotely. These are ordered top down. The first one matching will determine the oper's access. If access is denied, the command will be silently ignored.

The letters in parentheses denote the flags in `/stats U`.

shared {} variables

oper The user@host the oper must have, and the server they must be on. This may contain wildcards.

flags The list of what to allow, all the oper lines above this (up to another flags entry) will receive these flags. They are listed below.

> **Note:** While they have the same names, the flags have subtly different meanings from those in the cluster{} block.

shared {} flags

kline (K) Permanent and temporary `K:lines`

tkline (k) Temporary `K:lines`

unkline (U) `K:line` removals

xline (X) Permanent and temporary `X:lines`

txline (x) Temporary `X:lines`

unxline (Y) `X:line` removals

resv (Q) Permanently and temporarily reserved nicks/channels

tresv (q) Temporarily reserved nicks/channels

unresv (R) `RESV` removals

all All of the above; this does not include locops, rehash, dline, tdline or undline.

locops (L) `LOCOPS` messages (accepting this from * makes `LOCOPS` rather similar to `OPERWALL` which is not useful); unlike the other flags, this can only be accepted from `*@*` although it can be restricted based on source server.

rehash (H) `REHASH` commands; all options can be used

dline (D) Permanent and temporary `D:lines`

tdline (d) Temporary `D:lines`

undline (E) `D:line` removals

none Allow nothing to be done

service {} block

```
service {
    name = "text";
};
```

The service block specifies privileged servers (services). These servers have extra privileges such as setting login names on users and introducing clients with umode `+S` (unkickable, hide channels, etc). This does not allow them to set bans, you need a separate shared{} block for that.

Do not place normal servers here.

Multiple names may be specified but there may be only one service{} block.

service {} variables

name The server name to grant special privileges. This may not contain wildcards.

Hostname resolution (DNS)

Charybdis uses solely DNS for all hostname/address lookups (no `/etc/hosts` or anything else). The DNS servers are taken from `/etc/resolv.conf`. If this file does not exist or no valid IP addresses are listed in it, the local host (`127.0.0.1`) is used. (Note that the latter part did not work in older versions of Charybdis.)

IPv4 as well as IPv6 DNS servers are supported, but it is not possible to use both IPv4 and IPv6 in `/etc/resolv.conf`.

For both security and performance reasons, it is recommended that a caching nameserver such as BIND be run on the same machine as Charybdis and that `/etc/resolv.conf` only list `127.0.0.1`.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search