Vex-Cogs

Vexed

May 22, 2022

GETTING STARTED:

You can view documentation for each of my cogs below, or you can view the *Getting started* page if you want to install my cogs.

You can reach out to me in the Red - Cog Support server, in #support_vex-cogs.

CHAPTER

ONE

GETTING MY COGS

`[p]` is your prefix.

Note: You can replace `vex` with whatever you want, but you'll need to type it out every time you install one of my cogs so make sure it's simple. Note it's case sensitive.

1. **First, you need to add the my repository (repo):**

```
[p]repo add vex-cogs https://github.com/Vexed01/Vex-Cogs
```

2. **Now you can install my cogs with this command:**

```
[p]cog install vex-cogs cogname
```

3. **Finally, you need to load the cog:**

```
[p]load cogname
```

You can list my cogs with this command:

```
[p]cog list vex-cogs
```

Tip: It's a good idea to keep cogs up to date. You should run this command every now and again to update all your cogs:

```
[p]cog update
```

1.1 Cogs

Click a cog name to see detailed documentation.

Cog name	Summary
aliases	Get all the information you could ever need about a command's aliases.
anotherpingcog	Just another ping cog… But this one has fancy colours in an embed!
beautify	Beautify and minify JSON.
betteruptime	New uptime command that tracks the bot's uptime percentage (last 30 days).
birthday	Birthday cog with customisable messages and roles.
caseinsensitive	Make all prefixes and commands case insensitive.
cmdlog	Track command usage, searchable by user, server or command name.
covidgraph	Get graphs of COVID-19 data.
fivemstatus	View the live status of a FiveM server, in a updating Discord message.
github	Create, comment, labelify and close GitHub issues, with partial PR support.
googletrends	Find out what the world is searching, right from Discord.
madtranslate	Translate text through lots of languages. Get some funny results!
roleplay	Create an anonymous role play in your server.
stattrack	Track metrics about your bot and view them in Discord.
status	Recieve automatic status updates from various services, including Discord.
system	Get system metrics of the host device, such as RAM or CPU.
timechannel	Get the time in different timezones in voice channels.
wol	Use Wake on LAN from Discord! Sends magic packets on the local network.

1.2 Support

If you want some help with my cogs, you can ask in the Red - Cog Support server, in #support_vex-cogs.

CHAPTER

TWO

ANOTHERPINGCOG

This is the cog guide for the anotherpingcog cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

2.1 Usage

A rich embed ping command with latency timings.

You can customise the emojis, colours or force embeds with `[p]pingset`.

2.2 Commands

2.2.1 ping

Syntax

```
[p]ping
```

Description

A rich embed ping command with timings.

This will show the time to send a message, and the WS latency to Discord. If I can't send embeds or they are disabled here, I will send a normal message instead. The embed has more detail and is preferred.

2.2.2 pingset

Note: This command is locked to the bot owner.

Syntax

```
[p]pingset
```

Description

Manage settings - emojis, embed colour, and force embed.

pingset forceembed

Syntax

```
[p]pingset forceembed
```

Description

Toggle whether embeds should be forced.

If this is disabled, embeds will depend on the settings in `embedset`.

If it's enabled, embeds will embeds will always be sent unless the bot doesn't have permission to send them.

By default, this is True because the embed is richer and has more information. And it looks looks better.

This will be removed when a global per-command settings is available in Core Red.

pingset green

Syntax

```
[p]pingset green <emoji> [hex_colour=default]
```

Description

Set the colour and emoji to use for the colour Green.

If you want to go back to the defaults, just do `[p]pingset green default default`.

Arguments:

> `<emoji>` Just send the emoji as you normally would. It must be a custom emoji and I must be in the sever the emoji is in. You can also put `default` to use
>
> `[hex_colour]` (optional) The hex code you want the colour for Red to be. It looks best when this is the same colour as the emoji. Google "hex colour" if you need help with this.

Examples:

- `[p]pingset green :emoji:  #43B581`
- `[p]pingset green :emoji:  default`
- `[p]pingset green default #43B581`
- `[p]pingset green default default`

pingset orange

Syntax

```
[p]pingset orange <emoji> [hex_colour=default]
```

Description

Set the colour and emoji to use for the colour Orange.

If you want to go back to the defaults, just do `[p]pingset orange default default`.

Arguments:

> `<emoji>` Just send the emoji as you normally would. It must be a custom emoji and I must be in the sever the emoji is in. You can also put `default` to use
>
> `[hex_colour]` (optional) The hex code you want the colour for Red to be. It looks best when this is the same colour as the emoji. Google "hex colour" if you need help with this.

Examples:

- `[p]pingset orange :emoji:  #FAA61A`
- `[p]pingset orange :emoji:  default`
- `[p]pingset orange default #FAA61A`
- `[p]pingset orange default default`

pingset red

Syntax

```
[p]pingset red <emoji> [hex_colour=default]
```

Description

Set the colour and emoji to use for the colour Red.

If you want to go back to the defaults, just do `[p]pingset red default default`.

Arguments:

> `<emoji>` Just send the emoji as you normally would. It must be a custom emoji and I must be in the sever the emoji is in. You can also put `default` to use
>
> `[hex_colour]` (optional) The hex code you want the colour for Red to be. It looks best when this is the same colour as the emoji. Google "hex colour" if you need help with this.

Examples:

- `[p]pingset red :emoji:  #F04747`
- `[p]pingset red :emoji:  default`
- `[p]pingset red default #F04747`
- `[p]pingset red default default`

pingset settings

Syntax

```
[p]pingset settings
```

Description

See your current settings.

CHAPTER

THREE

ALIASES

This is the cog guide for the aliases cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

3.1 Usage

Get all the alias information you could ever want about a command.

3.2 Commands

3.2.1 aliases

Syntax

```
[p]aliases <command>
```

Description

Get all the alias information you could ever want about a command.

This will show the main command, built-in aliases, global aliases and server aliases.

Examples:

- `[p]aliases foo`
- `[p]aliases foo bar`

CHAPTER

FOUR

BEAUTIFY

This is the cog guide for the beautify cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

4.1 Usage

Beautify and minify JSON.

This cog has two commands, `[p]beautify` and `[p]minify`. Both of which behave in similar ways.

They are very flexible and accept inputs in many ways, for example replies or uploading - or just simply putting it after the command.

4.2 Commands

4.2.1 beautify

Syntax

```
[p]beautify [data]
```

Description

Beautify some JSON.

This command accepts it in a few forms.

1. Upload the JSON as a file (it can be .txt or .json) - Note that if you upload multiple files I will only scan the first one 2. Paste the JSON in the command - You send it raw, in inline code or a codeblock 3. Reply to a message with JSON - I will search for attachments and any codeblocks in the message

Examples:

- `[p]beautify {"1": "One", "2": "Two"}`
- `[p]beautify` (with file uploaded)
- `[p]beautify` (while replying to a messsage)

4.2.2 minify

Syntax

```
[p]minify [data]
```

Description

Minify some JSON.

This command accepts it in a few forms.

1. Upload the JSON as a file (it can be .txt or .json) - Note that if you upload multiple files I will only scan the first one 2. Paste the JSON in the command - You send it raw, in inline code or a codeblock 3. Reply to a message with JSON - I will search for attachments and any codeblocks in the message

Examples:

- `[p]minify {"1": "One", "2": "Two"}`
- `[p]minify` (with file uploaded)
- `[p]minify` (while replying to a messsage)

CHAPTER

FIVE

BETTERUPTIME

This is the cog guide for the betteruptime cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

5.1 Usage

Replaces the core `uptime` command to show the uptime percentage over the last 30 days.

The cog will need to run for a full 30 days for the full data to become available.

5.2 Commands

5.2.1 downtime

Syntax

```
[p]downtime [num_days=30]
```

Description

Check Red downtime over the last 30 days.

The default value for `num_days` is `30`. You can put `0` days for all-time data. Otherwise, it needs to be `5` or more.

Examples:

- `[p]uptime`
- `[p]uptime 0` (for all-time data)
- `[p]uptime 7`

5.2.2 uptime

Syntax

```
[p]uptime [num_days=30]
```

Description

Get Red's uptime percent over the last 30 days, and when I was last restarted.

The default value for `num_days` is `30`. You can put `0` days for all-time data. Otherwise, it needs to be `5` or more.

Examples:

- `[p]uptime`
- `[p]uptime 0` (for all-time data)
- `[p]uptime 7`

CHAPTER

SIX

BIRTHDAY

This is the cog guide for the birthday cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

6.1 Usage

Birthdays

Set yours and get a message and role on your birthday!

6.2 Commands

6.2.1 bdset

Note: This command is locked to server admins.

Syntax

```
[p]bdset
```

Description

Birthday management commands for admins.

Looking to set your own birthday? Use `[p]birthday set` or `[p]bday set`.

bdset channel

Syntax

```
[p]bdset channel <channel>
```

Description

Set the channel where the birthday message will be sent.

Example:

- `[p]bdset channel #birthdays` - set the channel to #birthdays

bdset interactive

Syntax

```
[p]bdset interactive
```

Description

Start interactive setup

bdset msgwithoutyear

Syntax

```
[p]bdset msgwithoutyear <message>
```

Description

Set the message to be send when the user provided a year.

Placeholders:

- `{name}` - the user's name
- `{mention}` - an @ mention of the user

All the placeholders are optional.

Examples:

- `[p]bdset msgwithoutyear Happy birthday {mention}!`
- `[p]bdset msgwithoutyear {mention}'s birthday is today! Happy birthday {name}.`

bdset msgwithyear

Syntax

```
[p]bdset msgwithyear <message>
```

Description

Set the message to be send when the user did provide a year.

Placeholders:

- `{name}` - the user's name
- `{mention}` - an @ mention of the user
- `{new_age}` - the user's new age

All the placeholders are optional.

Examples:

- `[p]bdset msgwithyear {mention} has turned {new_age}, happy birthday!`
- `[p]bdset msgwithyear {name} is {new_age} today! Happy birthday {mention}!`

bdset role

Syntax

```
[p]bdset role <role>
```

Description

Set the role that will be given to the user on their birthday.

You can give the exact name or a mention.

Example:

- `[p]bdset role @Birthday` - set the role to @Birthday
- `[p]bdset role Birthday` - set the role to @Birthday without a mention
- `[p]bdset role 418058139913063657` - set the role with an ID

bdset settings

Syntax

```
[p]bdset settings
```

Description

View your current settings

bdset time

Syntax

```
[p]bdset time <time>
```

Description

Set the time of day for the birthday message.

Minutes are ignored.

Examples:

- `[p]bdset time 7:00` - set the time to 7:45AM UTC
- `[p]bdset time 12AM` - set the time to midnight UTC

- `[p]bdset time 3PM` - set the time to 3:00PM UTC

6.2.2 birthday

Syntax

```
[p]birthday
```

Tip: Alias: `bday`

Description

Set and manage your birthday.

birthday remove

Syntax

```
[p]birthday remove
```

Tip: Aliases: `birthday delete`, `birthday del`

Description

Remove your birthday.

birthday set

Syntax

```
[p]birthday set <birthday>
```

Tip: Alias: `birthday add`

Description

Set your birthday.

You can optionally add in the year, if you are happy to share this.

If you use a date in the format xx/xx/xx or xx-xx-xx MM-DD-YYYY is assumed.

Examples:

- `[p]bday set 24th September`
- `[p]bday set 24th Sept 2002`
- `[p]bday set 9/24/2002`
- `[p]bday set 9-24-2002`
- `[p]bday set 9-24`

birthday upcoming

Syntax

```
[p]birthday upcoming [days=7]
```

Description

View upcoming birthdays.

Examples:

- `[p]birthday upcoming` - default of 7 days
- `[p]birthday upcoming 14` - 14 days

CHAPTER

SEVEN

CASEINSENSITIVE

This is the cog guide for the caseinsensitive cog. You will find detailed docs about usage and commands.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

7.1 Usage

This allows and commands to be case insensitive (for example `!Ping` would be accepted and responded to).

Whenever the cog is loaded, commands will be case insensitive. This cog itself has no commands.

If you want to disable it in a certain servers, use `[p]command disablecog CaseInsensitive`. There are also other configurations, such as setting a default as disabled and enabling per-server, listed under `[p]help command`.

7.2 Commands

This cog has no commands.

CHAPTER

EIGHT

CMDLOG

This is the cog guide for the cmdlog cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

8.1 Usage

Log command usage in a form searchable by user ID, server ID or command name.

The cog keeps an internal cache and everything is also logged to the bot's main logs under `red.vex.cmdlog`, level INFO.

8.2 Commands

8.2.1 cmdlog

Note: This command is locked to the bot owner.

Syntax

```
[p]cmdlog
```

Tip: Alias: `cmdlogs`

Description

View command logs.

Note the cache is limited to 100 000 commands, which is approximately 50MB of RAM

cmdlog cache

Syntax

```
[p]cmdlog cache
```

Description

Show the size of the internal command cache.

cmdlog command

Syntax

```
[p]cmdlog command <command>
```

Description

Upload all the logs that are stored for a specific command in the cache.

This does not check it is a real command, so be careful. Do not enclose it in ” if there are spaces.

You can search for a group command (eg `cmdlog`) or a full command (eg `cmdlog user`). As arguments are not stored, you cannot search for them.

Examples:

- `[p]cmdlog command ping`
- `[p]cmdlog command playlist`
- `[p]cmdlog command playlist create`

cmdlog full

Syntax

```
[p]cmdlog full
```

Description

Upload all the logs that are stored in the cache.

cmdlog server

Syntax

```
[p]cmdlog server <server_id>
```

Tip: Alias: `cmdlog guild`

Description

Upload all the logs that are stored for for a specific server ID in the cache.

Example:

- `[p]cmdlog server 527961662716772392`

cmdlog user

Syntax

```
[p]cmdlog user <user_id>
```

Description

Upload all the logs that are stored for a specific User ID in the cache.

Example:

- `[p]cmdlog user 418078199982063626`

CHAPTER

NINE

COVIDGRAPH

This is the cog guide for the covidgraph cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

9.1 Usage

Get COVID-19 graphs.

9.2 Commands

9.2.1 covidgraph

Syntax

```
[p]covidgraph
```

Description

Get graphs of COVID-19 data.

covidgraph cases

Syntax

```
[p]covidgraph cases [days] <country>
```

Tip: Alias: `covidgraph c`

Description

Get the number of confirmed cases in a country.

You can optionally specify the number of days to get data for, otherwise it will be all-time.

`country` can also be `world` to get the worldwide data.

Examples:

- `[p]covidgraph cases US` - All time data for the US
- `[p]covidgraph cases 7 US` - Last 7 days for the US
- `[p]covidgraph cases world` - Worldwide data

covidgraph deaths

Syntax

```
[p]covidgraph deaths [days] <country>
```

Tip: Alias: `covidgraph d`

Description

Get the number of deaths in a country.

You can optionally specify the number of days to get data for, otherwise it will be all-time.

`country` can also be `world` to get the worldwide data.

Examples:

- `[p]covidgraph deaths US` - All time data for the US
- `[p]covidgraph deaths 7 US` - Last 7 days for the US
- `[p]covidgraph deaths world` - Worldwide data

covidgraph vaccines

Syntax

```
[p]covidgraph vaccines [days] <country>
```

Tip: Alias: `covidgraph v`

Description

Get the number of vaccine doses administered in a country.

You can optionally specify the number of days to get data for, otherwise it will be all-time.

`country` can also be `world` to get the worldwide data.

Examples:

- `[p]covidgraph vaccines US` - All time data for the US
- `[p]covidgraph vaccines 7 US` - Last 7 days for the US
- `[p]covidgraph vaccines world` - Worldwide data

CHAPTER

TEN

FIVEMSTATUS

This is the cog guide for the fivemstatus cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

10.1 Usage

View the live status of a FiveM server, in a updating Discord message.

The message is an embed that updates minutely.

10.2 Commands

10.2.1 fivemstatus_loop

Note: This command is locked to server admins.

Syntax

```
[p]fivemstatus_loop
```

Description

Set up a live FiveM status embed.

To stop updating the message, just delete it.

fivemstatus_loop maintenance

Syntax

```
[p]fivemstatus_loop maintenance
```

Description

Toggle maintenance mode.

fivemstatus_loop setup

Syntax

```
[p]fivemstatus_loop setup <channel> <server>
```

Description

Set up a FiveM status message.

Examples:

- `[p]fivemstatus setup #status 0.0.0.0:30120`

CHAPTER

ELEVEN

GITHUB

This is the cog guide for the github cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

11.1 Usage

Create, comment, labelify and close GitHub issues.

This cog is only for bot owners. I made it for managing issues on my cog repo as a small project, but it certainly could be used for other situations where you want to manage GitHub issues from Discord.

If you would like a way to search or view issues, I highly recommend Kowlin's approved `githubcards` cog (on the repo https://github.com/Kowlin/Sentinel)

At present, this cannot support multiple repos.

PRs are mostly supported. You can comment on them or close them but not merge them or create them.

Get started with the `gh howtoken` command to set your GitHub token. You don't have to do this if you have already set it for a different cog, eg `githubcards`. Then set up with `gh setrepo`.

11.2 Commands

11.2.1 gh

Note: This command is locked to the bot owner.

Syntax

```
[p]gh
```

Tip: Alias: `github`

Description

Command to interact with this cog.

All commands are owner only.

gh addlabels

Syntax

```
[p]gh addlabels <issue>
```

Tip: Alias: `gh addlabel`

Description

Interactive command to add labels to an issue or PR.

gh close

Syntax

```
[p]gh close <issue>
```

Description

Close an issue or PR.

gh comment

Syntax

```
[p]gh comment <issue> <text>
```

Description

Comment on an issue or PR.

gh commentclose

Syntax

```
[p]gh commentclose <issue> <text>
```

Description

Comment on, then close, an issue or PR.

gh howtoken

Syntax

```
[p]gh howtoken
```

Description

Instructions on how to set up a token.

gh open

Syntax

```
[p]gh open <title>
```

Description

Open a new issue. Does NOT reopen.

gh removelabels

Syntax

```
[p]gh removelabels <issue>
```

Tip: Alias: `gh removelabel`

Description

Interactive command to remove labels from an issue or PR.

gh setrepo

Syntax

```
[p]gh setrepo <slug>
```

Description

Set up a repo to use as a slug (`USERNAME/REPO`).

CHAPTER

TWELVE

GOOGLETRENDS

This is the cog guide for the googletrends cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

12.1 Usage

Find what the world is searching, right from Discord.

Please note that there is no Google Trends API, so this is a web scraper and may break at any time.

12.2 Commands

12.2.1 trends

Syntax

```
[p]trends [timeframe=7d] [geo=world] <query...>
```

Description

Find what the world is searching, right from Discord.

Get started with ``[p]trends discord`` for a basic example!

Optional

`timeframe:` You can specify either the long (eg `4hours`) or short (`eg 4h`) version of the timeframes. All other values not listed below are invalid.

`hour/1h` `4hours/4h` `day/1d` `week/7d` `month/1m` `3months/3m` `12months/12m` `5years/5y` `all`

`geo:` Defaults to `world` You can specify a two-letter geographic code, such as `US`, `GB` or `FR`. Sometimes, you can also add a sub-region. See https://go.vexcodes.com/trends_geo for a list.

Required

trends: Whatever you want! You can add multiple trends, and separate them with a space. If your trend has spaces in it, you can use + instead of a space or enclose it in quotes, for example `Card games` to `Card+games` or `"Card games"`.

Examples:

- **`d;trends 1d US discord twitter youtube`** 1 day, United Stats searching for Discord, Twitter and YouTube.
- **`d;trends 1y COVID-19`** Trend for COVID-19 in the last year in the world
- **`d;trends all GB discord`** Trend for Discord in the United Kingdom for all time
- **`d;trends all US-NY "Donald Trump" "Joe Biden"`** A trend with spaces - Donald Trump and Joe Biden in New York State for all time

CHAPTER

THIRTEEN

MADTRANSLATE

This is the cog guide for the madtranslate cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

13.1 Usage

Translate things into lots of languages then back to English!

This will defiantly have some funny moments… Take everything with a pinch of salt!

13.2 Commands

13.2.1 madtranslate

Syntax

```
[p]madtranslate [count=15] <text_to_translate>
```

Tip: Aliases: `mtranslate`, `mtrans`

Description

Translate something into lots of languages, then back to English!

Examples:

- `[p]mtrans This is a sentence.`
- `[p]mtrans 25 Here's another one.`

At the bottom of the output embed is a count-seed pair. You can use this with the `mtransseed` command to use the same language set.

13.2.2 mtransseed

Syntax

```
[p]mtransseed <count_seed> <text_to_translate>
```

Description

Use a count-seed pair to (hopefully) get reproducible results.

They may be unreproducible if Google Translate changes its translations.

The count-seed pair is obtained from the main command, `mtrans`, in the embed footer.

Examples:

- `[p]mtrans 15-111111 This is a sentence.`
- `[p]mtrans 25-000000 Here's another one.`

CHAPTER

FOURTEEN

ROLEPLAY

This is the cog guide for the roleplay cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

14.1 Usage

Set up a role play, where the author of messages are secret - the bot reposts all messages.

Admins can get started with `[p]roleplay channel`, as well as some other configuration options.

14.2 Commands

14.2.1 roleplay

Syntax

```
[p]roleplay
```

Description

Role play configuration.

This is a group command, so you can use it to configure the roleplay for a channel.

Get started with `[p]roleplay channel`.

roleplay channel

Syntax

```
[p]roleplay channel [channel]
```

Description

Set the channel for the roleplay.

Leave blank to disable.

Examples:

- `[p]roleplay channel` - disable roleplay
- `[p]roleplay channel #roleplay` - set the channel to #roleplay

roleplay embed

Syntax

```
[p]roleplay embed <embed>
```

Description

Enable or disable embeds.

The default is disabled.

Examples:

- `[p]roleplay embed true` - enable
- `[p]roleplay embed false` - disable

roleplay log

Syntax

```
[p]roleplay log [channel]
```

Description

Set a channel to log role play messages to.

If you do not specify a channel logging will be disabled.

Examples:

- `[p]roleplay log #logs` - set to a channel called logs
- `[p]roleplay log` - disable logging

roleplay radio

Syntax

```
[p]roleplay radio <radio>
```

Description

Enable or disable radio.

The default is disabled.

Examples:

- `[p]roleplay radio true` - enable radio mode
- `[p]roleplay radio false` - disable radio mode

roleplay settings

Syntax

```
[p]roleplay settings
```

Description

View the current settings for the roleplay.

CHAPTER

FIFTEEN

STATTRACK

This is the cog guide for the stattrack cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

15.1 Resource Usage

CPU usage depends on your bot size and host machine performance. Because this is Python, the cog can (for the most part) only use one core. You can check the performance of the background loop with *stattrackinfo*.

For disk usage, this cog uses around 150KB per day. This is just around 50MB per year (the cog will NOT automatically delete old data so this will increase over time) It uses an SQLite database that requires no extra setup.

RAM usage will be at least double disk usage and may spike to more when commands are used or the loop is active.

15.2 Usage

Track your bot's metrics and view them in Discord.

Commands will output as a graph. Data can also be exported with `[p]stattrack export` into a few different formats.

15.3 Commands

15.3.1 stattrack

Syntax

```
[p]stattrack
```

Description

View my stats.

stattrack channels

Syntax

```
[p]stattrack channels [timespan=1d] [metrics]
```

Description

Get channel stats.

You can just run this command on its own to see all metrics, or specify some metrics - see below.

Arguments

`[timespan]` How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

`[metrics]` The metrics to show. Valid options: `total`, `text`, `voice`, `stage`, `category`. Defaults to all of them.

Note that `total` will count users multiple times if they share multiple servers with the Red, while `unique` will only count them once.

Examples:

- `[p]stattrack servers 3w2d`
- `[p]stattrack servers 5d`
- `[p]stattrack servers all`

stattrack commands

Syntax

```
[p]stattrack commands [timespan=1 day, 0:00:00]
```

Description

Get command usage stats.

Arguments

<timespan> How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

Examples:

- `[p]stattrack commands 3w2d`
- `[p]stattrack commands 5d`
- `[p]stattrack commands all`

stattrack export

Note: This command is locked to the bot owner.

Syntax

```
[p]stattrack export
```

Description

Export stattrack data.

stattrack export csv

Syntax

```
[p]stattrack export csv
```

Description

Export as CSV

stattrack export json

Syntax

```
[p]stattrack export json
```

Description

Export as JSON with pandas orient “split”

stattrack latency

Syntax

```
[p]stattrack latency [timespan=1 day, 0:00:00]
```

Tip: Alias: `stattrack ping`

Description

Get my latency stats.

Arguments

<timespan> How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

Examples:

- `[p]stattrack latency 3w2d`
- `[p]stattrack latency 5d`
- `[p]stattrack latency all`

stattrack looptime

Syntax

```
[p]stattrack looptime [timespan=1 day, 0:00:00]
```

Tip: Aliases: `stattrack time`, `stattrack loop`

Description

Get my loop time stats.

Arguments

<timespan> How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

Examples:

- `[p]stattrack looptime 3w2d`
- `[p]stattrack looptime 5d`
- `[p]stattrack looptime all`

stattrack messages

Syntax

```
[p]stattrack messages [timespan=1 day, 0:00:00]
```

Description

Get message stats.

Arguments

<timespan> How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

Examples:

- `[p]stattrack messages 3w2d`
- `[p]stattrack messages 5d`
- `[p]stattrack messages all`

stattrack servers

Syntax

```
[p]stattrack servers [timespan=1 day, 0:00:00]
```

Tip: Alias: `stattrack guilds`

Description

Get server stats.

Arguments

<timespan> How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

Examples:

- `[p]stattrack servers 3w2d`
- `[p]stattrack servers 5d`
- `[p]stattrack servers all`

stattrack status

Syntax

```
[p]stattrack status [timespan=1d] [metrics]
```

Description

Get status stats.

You can just run this command on its own to see all metrics, or specify some metrics - see below.

Arguments

`[timespan]` How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

`[metrics]` The metrics to show. Valid options: `online`, `idle`, `offline`, `dnd`. Defaults to all of them.

Examples:

- `[p]stattrack status` - show all metrics, 1 day
- `[p]stattrack status 3w2d` - show all metrics, 3 weeks 2 days
- `[p]stattrack status 5d dnd online` - show dnd & online, 5 days
- `[p]stattrack status all online idle` - show online & idle, all time

stattrack system

Syntax

```
[p]stattrack system
```

Tip: Alias: `stattrack sys`

Description

Get system metrics.

stattrack system cpu

Syntax

```
[p]stattrack system cpu [timespan=1 day, 0:00:00]
```

Description

Get CPU stats.

Arguments

<timespan> How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

Examples:

- `[p]stattrack system cpu 3w2d`
- `[p]stattrack system cpu 5d`
- `[p]stattrack system cpu all`

stattrack system mem

Syntax

```
[p]stattrack system mem [timespan=1 day, 0:00:00]
```

Tip: Aliases: `stattrack system memory`, `stattrack system ram`

Description

Get memory usage stats.

Arguments

<timespan> How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

Examples:

- `[p]stattrack system mem 3w2d`
- `[p]stattrack system mem 5d`
- `[p]stattrack system mem all`

stattrack users

Syntax

```
[p]stattrack users [timespan=1d] [metrics]
```

Description

Get user stats.

You can just run this command on its own to see all metrics, or specify some metrics - see below.

Arguments

`[timespan]` How long to look for, or `all` for all-time data. Defaults to 1 day. Must be at least 1 hour.

`[metrics]` The metrics to show. Valid options: `total`, `unique`, `humans`, `bots`. Defaults to all of them.

Note that `total` will count users multiple times if they share multiple servers with the Red, while `unique` will only count them once.

Examples:

- `[p]stattrack users` - show all metrics, 1 day
- `[p]stattrack users 3w2d` - show all metrics, 3 weeks 2 days
- `[p]stattrack users 5d total unique` - show total & unique, 5 days
- `[p]stattrack users all humans bots` - show humans & bots, all time

CHAPTER

SIXTEEN

STATUS

This is the cog guide for the status cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

16.1 Usage

Automatically check for status updates.

When there is one, it will send the update to all channels that have registered to recieve updates from that service.

There's also the `status` command which anyone can use to check updates whereever they want.

If there's a service that you want added, contact Vexed#9000 or make an issue on the GitHub repo (or even better a PR!).

16.2 Commands

16.2.1 status

Syntax

```
[p]status <service>
```

Description

Check for the status of a variety of services, eg Discord.

Example:

- `[p]status discord`

16.2.2 statusset

Note: This command is locked to server admins.

Syntax

```
[p]statusset
```

Description

Get automatic status updates in a channel, eg Discord.

Get started with `[p]statusset preview` to see what they look like, then `[p]statusset add` to set up automatic updates.

statusset add

Syntax

```
[p]statusset add <service> [channel]
```

Description

Start getting status updates for the chosen service!

There is a list of services you can use in the `[p]statusset list` command.

This is an interactive command. It will ask what mode you want to use and if you want to use a webhook. You can use the `[p]statusset preview` command to see how different options look or take a look at https://go.vexcodes.com/c/statusref

If you don't specify a specific channel, I will use the current channel.

statusset edit

Syntax

```
[p]statusset edit
```

Description

Edit services you've already set up.

statusset edit mode

Syntax

```
[p]statusset edit mode [channel] <service> <mode>
```

Description

Change what mode to use for status updates.

All: Every time the service posts an update on an incident, I will send a new message containing the previous updates as well as the new update. Best used in a fast-moving channel with other users.

Latest: Every time the service posts an update on an incident, I will send a new message containing only the latest update. Best used in a dedicated status channel.

Edit: When a new incident is created, I will sent a new message. When this incident is updated, I will then add the update to the original message. Best used in a dedicated status channel.

If you don't specify a channel, I will use the current channel.

Examples:

- `[p]statusset edit mode #testing discord latest`
- `[p]statusset edit mode discord edit` (for current channel)

statusset edit restrict

Syntax

```
[p]statusset edit restrict [channel] <service> <restrict>
```

Description

Restrict access to the service in the `status` command.

Enabling this will reduce spam. Instead of sending the whole update (if there's an incident) members will instead be redirected to channels that automatically receive the status updates, that they have permission to to view.

Examples:

- `[p]statusset edit restrict #testing discord true`
- `[p]statusset edit restrict discord false` (for current channel)

statusset edit webhook

Syntax

```
[p]statusset edit webhook [channel] <service> <webhook>
```

Description

Set whether or not to use webhooks for status updates.

Using a webhook means that the status updates will be sent with the avatar as the service's logo and the name will be `[service] Status Update`, instead of my avatar and name.

If you don't specify a channel, I will use the current channel.

Examples:

- `[p]statusset edit webhook #testing discord true`
- `[p]statusset edit webhook discord false` (for current channel)

statusset list

Syntax

```
[p]statusset list [service]
```

Tip: Aliases: `statusset show`, `statusset settings`

Description

List that available services and ones are used in this server.

Optionally add a service at the end of the command to view detailed settings for that service.

Examples:

- `[p]statusset list discord`
- `[p]statusset list`

statusset preview

Syntax

```
[p]statusset preview <service> <mode> <webhook>
```

Description

Preview what status updates will look like.

You can also see this at https://go.vexcodes.com/c/statusref

<service>

The service you want to preview. There's a list of available services in the `[p]help statusset` command.

<mode>

all: Every time the service posts an update on an incident, I will send a new message containing the previous updates as well as the new update. Best used in a fast-moving channel with other users.

latest: Every time the service posts an update on an incident, I will send a new message containing only the latest update. Best used in a dedicated status channel.

edit: Naturally, edit mode can't have a preview so won't work with this command. The message content is the same as the `all` mode. When a new incident is created, I will sent a new message. When this incident is updated, I will then add the update to the original message. Best used in a dedicated status channel.

<webhook>

Using a webhook means that the status updates will be sent with the avatar as the service's logo and the name will be `[service] Status Update`, instead of my avatar and name.

Examples:

- `[p]statusset preview discord all true`
- `[p]statusset preview discord latest false`

statusset remove

Syntax

```
[p]statusset remove <service> [channel]
```

Tip: Aliases: `statusset del`, `statusset delete`

Description

Stop status updates for a specific service in this server.

If you don't specify a channel, I will use the current channel.

Examples:

- `[p]statusset remove discord #testing`
- `[p]statusset remove discord` (for using current channel)

CHAPTER

SEVENTEEN

STATUS REFERENCE

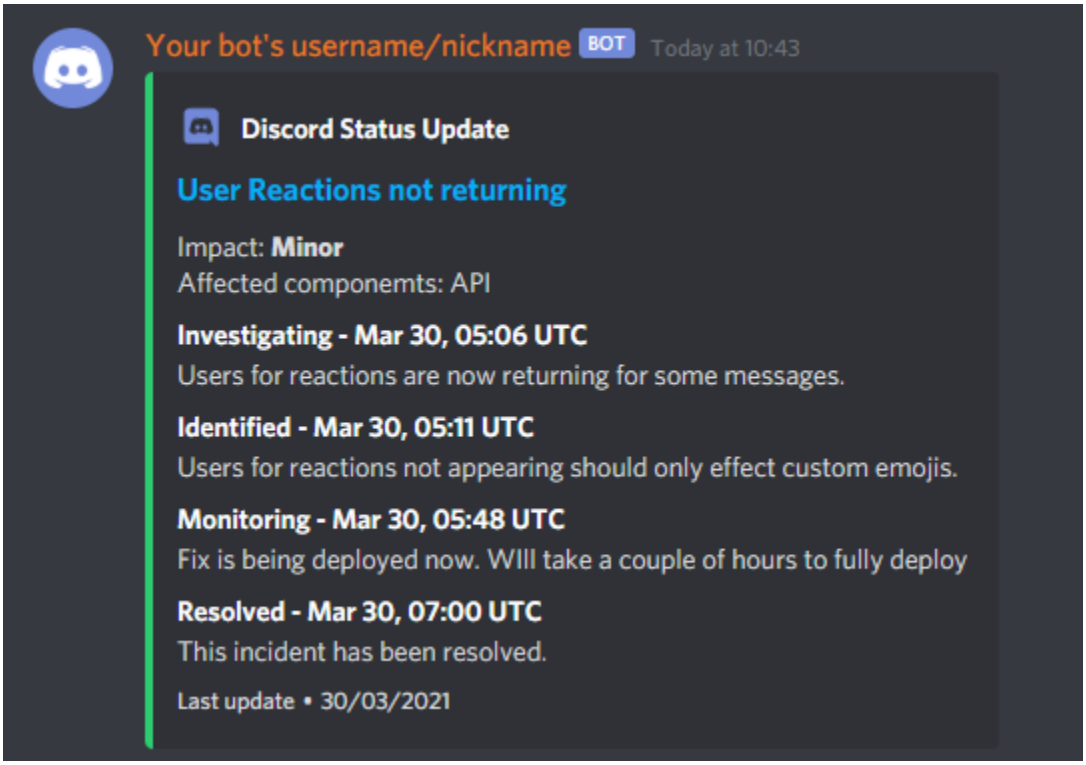
Below you will see previews for different modes, and webhook.

17.1 Modes

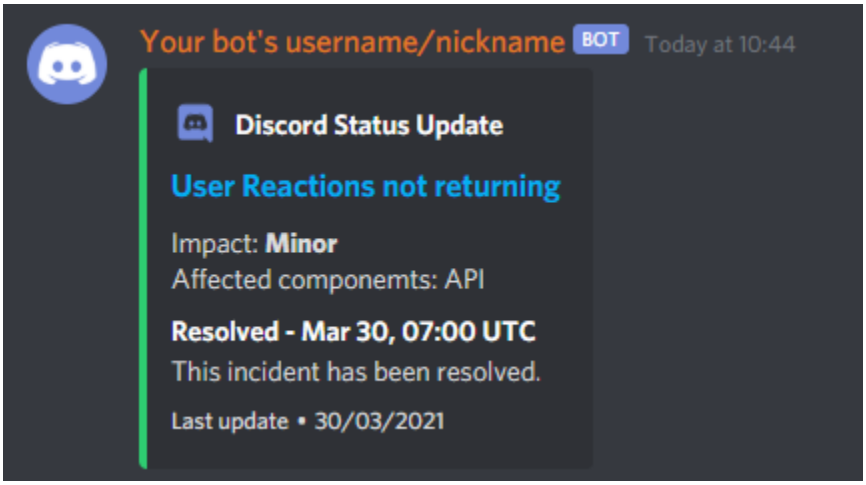
The below modes were sent *without* a webhook.

17.1.1 All and Edit

Note: The edit mode is the same as the all mode, however it will only send one message per per incident. This message will then be edited to reflect recent updates.

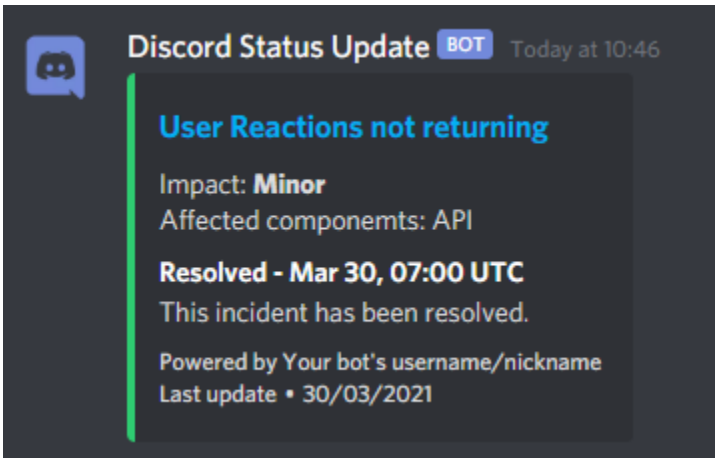


17.1.2 Latest



17.2 Webhook

To stay brief only the `latest` mode is included, however the `all` and `edit` modes are also fully supported - just with a few more fields in the embed.



CHAPTER

EIGHTEEN

SYSTEM

This is the cog guide for the system cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

18.1 Usage

Get system metrics.

Most commands work on all OSes or omit certian information. See the help for individual commands for detailed limitations.

18.2 Commands

18.2.1 system

Note: This command is locked to the bot owner.

Syntax

```
[p]system
```

Description

Get information about your system metrics.

Most commands work on all OSes or omit certian information. See the help for individual commands for detailed limitations.

system cpu

Syntax

```
[p]system cpu
```

Description

Get metrics about the CPU.

This will show the CPU usage as a percent for each core, and frequency depending on platform. It will also show the time spent idle, user and system as well as uptime.

Platforms: Windows, Linux, Mac OS .. Note:: CPU frequency is nominal and overall on Windows and Mac OS,

on Linux it's current and per-core.

system disk

Syntax

```
[p]system disk
```

Tip: Alias: `system df`

Description

Get infomation about disks connected to the system.

This will show the space used, total space, filesystem and mount point (if you're on Linux make sure it's not potentially sensitive if running the command a public space).

Platforms: Windows, Linux, Mac OS

Note: Mount point is basically useless on Windows as it's the same as the drive name, though it's still shown.

system mem

Syntax

```
[p]system mem
```

Tip: Aliases: `system memory`, `system ram`

Description

Get infomation about memory usage.

This will show memory available as a percent, memory used and available as well as the total amount. Data is provided for both physical and SWAP RAM.

Platforms: Windows, Linux, Mac OS

system network

Syntax

```
[p]system network
```

Tip: Alias: `system net`

Description

Get network stats. They may have overflowed and reset at some point.

Platforms: Windows, Linux, Mac OS

system processes

Syntax

```
[p]system processes
```

Tip: Alias: `system proc`

Description

Get an overview of the status of currently running processes.

Platforms: Windows, Linux, Mac OS

system sensors

Syntax

```
[p]system sensors [fahrenheit=False]
```

Tip: Aliases: `system temp`, `system temperature`, `system fan`, `system fans`

Description

Get sensor metrics.

This will return any data about temperature and fan sensors it can find. If there is no name for an individual sensor, it will use the name of the group instead.

Platforms: Linux

system top

Syntax

```
[p]system top
```

Tip: Aliases: `system overview`, `system all`

Description

Get an overview of the current system metrics, similar to `top`.

This will show CPU utilisation, RAM usage and uptime as well as active processes.

Platforms: Windows, Linux, Mac OS

Note: This command appears to be very slow in Windows.

system uptime

Syntax

```
[p]system uptime
```

Tip: Alias: `system up`

Description

Get the system boot time and how long ago it was.

Platforms: Windows, Linux, Mac OS

system users

Syntax

```
[p]system users
```

Description

Get information about logged in users.

This will show the user name, what terminal they're logged in at, and when they logged in.

Platforms: Windows, Linux, Mac OS

Note: PID is not available on Windows. Terminal is usually `Unknown`

CHAPTER

NINETEEN

TIMECHANNEL

This is the cog guide for the timechannel cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

19.1 Usage

Allocate a Discord voice channel to show the time in specific timezones. Updates every hour.

A list of timezones can be found here, though you should be able to enter any major city: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones#List

There is a fuzzy search so you don't need to put the region in, only the city.

The `[p]timezones` command (runnable by anyone) will show the full location name.

19.2 Commands

19.2.1 timechannelset

Note: This command is locked to server admins.

Syntax

```
[p]timechannelset
```

Tip: Alias: `tcset`

Description

Manage channels which will show the time for a timezone.

timechannelset create

Syntax

```
[p]timechannelset create <string>
```

Description

Set up a time channel in this server.

If you move the channel into a category, **click 'Keep Current Permissions' in the sync permissions dialogue.**

How to use this command:

First, use the `[p]tcset short <long_tz>` to get the short identifier for the timezone of your choice.

Once you've got a short identifier from `tcset short`, you can use it in this command. Simply put curly brackets, `{` and `}` around it, and it will be replaced with the time.

For example, running `[p]tcset short new york` gives a short identifier of `fv`. This can then be used like so: `[p]tcset create New York: {fv}`.

You could also use two in one, for example `[p]tcset create UK: {ni} FR: {nr}`

The default is 12 hour time, but you can use `{shortid-24h}` for 24 hour time, eg `{ni-24h}`

More Examples:

- `[p]tcset create New York: {fv}`
- `[p]tcset create UTC: {qw}`
- `[p]tcset create {ni-24h} in London`
- `[p]tcset create US Pacific: {qv-24h}`

timechannelset remove

Syntax

```
[p]timechannelset remove <channel>
```

Description

Delete and stop updating a channel.

For the <channel> argument, you can use its ID or mention (type #!channelname)

Example:

- `[p]tcset remove #!channelname` (the ! is how to mention voice channels)
- `[p]tcset remove 834146070094282843`

timechannelset short

Syntax

```
[p]timechannelset short <timezone>
```

Description

Get the short identifier for the main `create` command.

The list of acceptable timezones is here (the "TZ database name" column): https://en.wikipedia.org/wiki/List_of_tz_database_time_zones#List

There is a fuzzy search, so you shouldn't need to enter the region.

Please look at `[p]help tcset create` for more information.

Examples:

- `[p]tcset short New York`
- `[p]tcset short UTC`
- `[p]tcset short London`
- `[p]tcset short Europe/London`

19.2.2 timezones

Syntax

```
[p]timezones
```

Description

See the time in all the configured timezones for this server.

CHAPTER

TWENTY

UPTIMERESPONDER

This is the cog guide for the uptimeresponder cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

20.1 Usage

A cog for responding to pings form various uptime monitoring services, such as UptimeRobot, Pingdom, Uptime.com, or self-hosted ones like UptimeKuma or Upptime.

The web server will run in the background whenever the cog is loaded on the specified port.

20.2 Commands

20.2.1 uptimeresponderport

Note: This command is locked to the bot owner.

Syntax

```
[p]uptimeresponderport [port]
```

Description

Get or set the port to run the simple web server on.

Run the command on it's own (`[p]uptimeresponderport`) to see what it's set to at the moment, and to set it run `[p]uptimeresponderport 8080`, for example.

CHAPTER

TWENTYONE

WOL

This is the cog guide for the wol cog. You will find detailed docs about usage and commands.

`[p]` is considered as your prefix.

Note: To use this cog, you will need to install and load it.

See the *Getting my cogs* page.

21.1 Usage

Send a magic packet (Wake on LAN) to a computer on the local network.

Get started by adding your computer with `[p]wolset add <friendly_name> <mac>`. Then you can wake it with `[p]wol <friendly_name>`.

For example, `[p]wolset add main_pc 11:22:33:44:55:66` then you can use `[p]wol main_pc`

21.2 Commands

21.2.1 wol

Note: This command is locked to the bot owner.

Syntax

```
[p]wol <machine>
```

Description

Wake a local computer.

You can set up a short name with `[p]wolset add` so you don't need to write out the MAC each time, or just send the MAC.

Examples:

- `[p]wol main_pc`
- `[p]wol 11:22:33:44:55:66`

21.2.2 wolset

Note: This command is locked to the bot owner.

Syntax

```
[p]wolset
```

Description

Manage your saved computer/MAC aliases for easy access.

wolset add

Syntax

```
[p]wolset add <friendly_name> <mac>
```

Description

Add a machine for easy use with `[p]wol`.

`<friendly_name>` **cannot** include spaces.

Examples:

- `wolset add main_pc 11:22:33:44:55:66`
- `wolset add main_pc 11-22-33-44-55-66`

wolset list

Syntax

```
[p]wolset list
```

Description

See your added addresses.

This will send your MAC addresses to current channel.

wolset remove

Syntax

```
[p]wolset remove <friendly_name>
```

Tip: Aliases: `wolset del`, `wolset delete`

Description

Remove a machine from my list of machines.

Examples:

- wolset remove main_pc

CHAPTER

TWENTYTWO

STATUS EVENTS

The status cog has two events you can listen to, `on_vexed_status_update` and `on_vexed_status_channel_send`.

`status_channel_send` is fired in quick succession, especially on larger bots with lots of channels added, so you shouldn't do anything expensive. `status_channel_send` is dispatched after a successful channel send, so it won't be dispatched if the bot couldn't send for whatever reason.

`status_update` is dispatched with the channels the cog intends to send updates to when an update has been confirmed as a non-ghost update.

There is a guaranteed delay of 1 second between `status_update` and the first `status_channel_send` to allow you to perform an expensive process (or avoid repeated config calls for each dispatch) and then cache the result for the when `status_channel_send` dispatches for each channel so you get the timing correct and you can guarantee it was sent.

Though this is incredibly unlikely, the cog will cancel sending updates (and the subsequent `status_channel_send`) if it lasts longer than 4 minutes after it started that check for updates. Note multiple services' updates may be included in this time.

The events are linear. `on_status_update` guarantees the next `status_channel_send` will be the same update.

Note: If you are using this cog/event to get a parsed update to send yourself, note that `status_update` will not trigger if no channels are subscribed to the service - the cog only checks feeds that have channels subscribed to it.

Tip: For testing, the `statusdev checkfeed` (alias `statusdev cf`) command can be used. It will send the latest incident and scheduled maintenance for the service provided to the current channel, with the `force` parameter being `True`.

You can also use `statusdev forcestatus` (alias `statusdev fs`) which will send the latest incident to ALL channels in ALL servers that recieve that service's incidents.

22.1 Example

```
@commands.Cog.listener()
async def on_vexed_status_update(self, **_kwargs):
    data = await self.config.all_channels()  # get you data from config here
    self.data_cache = data

@commands.Cog.listener()
async def on_vexed_status_channel_send(self, *, service, channel_data, **_kwargs):
```

(continues on next page)

(continued from previous page)

```
    data = self.data_cache.get(channel_data.channel.id)
    # then get it back here for each channel send to reduce config calls,
    # esp on larger bots

    if data is None:
        return

    mention_ids = data["user_mentions"].get(service)
    # if you registered in config as user_mentions
    if mention_ids is None:
        return

    mention_ids = [f"<@{id}>" for id in mention_ids]
    await channel_data.channel.send(humanize_list(mention_ids))
```

22.2 Event Reference

on_vexed_status_update(*update*, *service*, *channels*, *force*)

This event triggers before updates are sent to channels. See above for details.

Parameters

- **update** (Update) – The main class with the update information, including what was sent and the settings it was sent with. It has subclasses in the attributes - see below *Custom Objects*.
- **service** (str) – The name of the service, in lower case. Guaranteed to be on of the keys in the file-level consts of `status.py`, though new services are being added over time so don't copy-paste and expect it to be one of them.
- **channels** (dict) – A dict with the keys as channel IDs and the values as a nested dict containing the settings for that channel.
- **force** (bool) – Whether or not the update was forced to update with `statusdev checkfeed/statusdev cf`

on_vexed_status_channel_send(*update*, *service*, *channel_data*, *force*)

This is has similarities to the above event, mainly that it dispatches after an update was successfully sent to a specific channel. See above info at the top of this page for details.

Parameters

- **update** (Update) – The main class with the update information, including what was sent and the settings it was sent with. It has subclasses - see below *Custom Objects*.
- **service** (str) – The name of the service, in lower case. Guaranteed to be on of the keys in the file-level consts of `status.py`, though new services are being added over time so don't copy-paste and expect it to be one of them.
- **channel_data** (ChannelData) – The channel it was sent to and the associated settings. It has subclasses in the attributes - see below *Custom Objects*.
- **force** (bool) – Whether or not the update was forced to update with `statusdev checkfeed/statusdev cf`

22.3 Custom Objects

22.3.1 ChannelData

`objects/channel.py` (ignore the custom errors in this file) This object has all the settings that the update was sent with.

Attributes

channel (`discord.TextChannel`) – Idk, this might just be the channel the update was sent to.
mode (`str`) – The mode the update was sent as.
webhook (`bool`) – Whether or not it was sent as a webhook.
edit_id (`Dict[str, int]`) – I cba to explain this, you don't need to know.
embed (`bool`) – Whether or not it was sent as an embed.

22.3.2 Update

`objects/incidentdata.py` This is a base object from which the two below are nested in.

Attributes

incidentdata (`incidentdata`) – The feed data from which the update was sent. See below.
new_fields (`List[UpdateField]`) – A list of new fields since the service was last checked. Usually 1.

22.3.3 incidentdata

`objects/incidentdata.py` This is present in the `incidentdata` attribute of the `Update` object.

Attributes

fields (`List[UpdateField]`) – A list containing UpdateField objects
title (`str`) – The title of the incident
time: (`datetime` | None) - Parsed time, or None
link (`str`) – The incident link.
actual_time: (`datetime` | None) - Parsed time, or None
description (`str` | None) – Exclusively used for when a scheduled incident is being sent
incident_id (`str`) – The incident's unique ID
scheduled_for: (`datetime` | None) If the incident sent was scheduled, this is when the event starts/started

Methods

to_dict() – Get a dict of the data held in the object
get_update_ids() – Get the group IDs. These are unique and represent each update. See UpdateField for more information

22.3.4 UpdateField

`objects/incidentdata.py` This is present in the `fields` attribute of the above `incidentdata` object and the `new_fields` attribute of the `Update` object.

Attributes

name (`str`) – The name of the field
value (`str`) – The value of the field
update_id (`str`) – The group ID of the field. These are unique unless the field was split up to accommodate embed limits

CHAPTER

TWENTYTHREE

CHANGELOG

I may sometimes push an update without incrementing the version. These will not be put in the changelog.

Usage of this for all version bumping updates started 21-04-08.

Date format throughout is YYYY-MM-DD

Jump links:

Note: Changelogs are automaticity generated. As such, there may sometimes be visual glitches as I do not check this.

CHAPTER

TWENTYFOUR

ALIASES

24.1 1.0.6

2022-01-15

- Show correct command name

24.2 1.0.5

2021-08-24

- Add opt-in telemetry and error reporting

24.3 1.0.4

2021-04-11

- Fix edge case to hide alias cog aliases if they are a built in command/command alias

24.4 1.0.3

2021-04-08

- Fix logic for checking command
- Small internal cleanup (still more to do)

CHAPTER

TWENTYFIVE

ANOTHERPINGCOG

25.1 1.1.7

2021-10-04

- Fix OverflowError in edge cases (ANOTHERPINGCOG-2 on Sentry)

25.2 1.1.6

2021-08-24

- Add opt-in telemetry and error reporting

25.3 1.1.5

2021-07-18

- Allow customisation of embed footer (#35 by Obi-Wan3)

25.4 1.1.4

2021-05-09

- Potentially fix super edge case behaviour with command not registering

CHAPTER

TWENTYSIX

BEAUTIFY

26.1 1.1.2

2021-08-24

- Add opt-in telemetry and error reporting

26.2 1.1.1

2021-04-24

- Internal: switch to `pyjson5.decode` instead of `pyjson5.loads`

26.3 1.1.0

2021-04-21

26.3.1 User-facing changes

- Accept more values (True, False and None in that specific casing)

26.3.2 Internal Changes

- Cache whether pyjson5 is available instead of catching NameError each time
- Move more stuff to utils to better apply DRY

26.4 1.0.3

2021-04-21

- Add EUD key to `__init__.py`

26.5 1.0.2

2021-04-12

- Remove print statement
- Allow py codeblocks in replies (eg for beautifying an eval)

26.6 1.0.1

2021-04-12

- Use JSON5 to support Python dicts

26.7 1.0.0

2021-04-11

- Initial release

CHAPTER

TWENTYSEVEN

BETTERUPTIME

27.1 2.1.3

2022-02-07

- Fix log error in uptime graph

27.2 2.1.2

2021-11-09

- Fix incorrect percentages in graph annotation

27.3 2.1.1

2021-11-09

- Limit annotated points on uptime graph to 5

27.4 2.1.0

2021-11-09

- Move plotting backend to Plotly

27.5 2.0.6

2021-09-14

- Theoretically fix plotting error in certian situations

27.6 2.0.5

2021-08-24

- Add opt-in telemetry and error reporting

27.7 2.0.4

2021-08-11

- Fix edge case KeyError

27.8 2.0.3

2021-07-28

- Use Discord's new timestamp format

27.9 2.0.2

2021-06-21

- Add labels to uptime under 99.7% to graph

27.10 2.0.1

2021-06-21

- Require 4+ days of data for graph

27.11 2.0.0

2021-06-21

- Significant internal refactoring to make it more maintainable
- New command: `uptimegraph` - see uptime in graph form
- New command: `uptimeexport` (bot owner only) - export uptime data to CSV
- Fix removing wrong command on cog unload

27.12 1.6.0

2021-06-06

- Add *resetbu* command to reset all uptime data

27.13 1.6.0

2021-05-28

- Fix commands
- Fix config migration

27.14 1.5.2

2021-05-25

- Remove custom uptime command… There's some broken shit that I can't fix, rewrite was already planned and this will be fixed then (#23 on GitHub)

27.15 1.5.1

2021-05-23

- Fix deprecation warning

27.16 1.5.0

2021-05-23

- Move to storing and internally cache data as a Pandas Series

27.17 1.4.1

2021-05-09

- Fix unreachable code

27.18 1.4.0

2021-05-01

- Utilise an Abstract Base Class and move to VexLoop

27.19 1.3.0

2021-04-25

- Allow a custom timeframe in `uptime` and `downtime`, eg `uptime 7`
- Pagify the `downtime` command

27.20 1.2.2

- Slight logic changes for banding in `downtime` command

CHAPTER

TWENTYEIGHT

BIRTHDAY

28.1 1.1.1

2022-05-20

- Show correct age a user will turn in *birthday upcoming* when their next birthday is next year

28.2 1.1.0

2022-04-14

- Add command to stop messages & roles in a guild
- Add dev env value

28.3 1.0.12

2022-04-06

- Don't allow users to set birthday messages which contain invalid placeholders (GH #92)

28.4 1.0.11

2022-03-24

- Fix bug in `bdset interactive` where setup continues after a timeout, causing some issues
- Fix bug in `bdset settings` where time could be None (likely caused by ^)

28.5 1.0.10

2022-02-18

- Fix birthday role logic again

28.6 1.0.9

2022-02-16

- Fix role perm check

28.7 1.0.8

2022-02-15

- Add warnings to `bdset settings` if channels or roles are incorrectly configured
- Modify internal logic for checking for channel and role perms

28.8 1.0.7

2022-02-08

- More extensive permission checks

28.9 1.0.6

2022-02-08

- Ensure announcements are on the correct day when a non-UTC midnight time is used v2

28.10 1.0.5

2022-02-07

- Ensure announcements are on the correct day when a non-UTC midnight time is used

28.11 1.0.4

2022-02-06

- Grab the config instance instead of json (#79)

28.12 1.0.3

2022-02-06

- Catch OverflowError in *bdset zemigrate*

28.13 1.0.2

2022-02-05

- Add `[p]bdset zemigrate` for migrating data from ZeLarp's/flare's fork of Birthdays cog (#77)

28.14 1.0.1

2022-02-05

- Add `[p]bdset force` for admins to force set a user's birthday

28.15 1.0.0

- Initial release

CHAPTER

TWENTYNINE

CASEINSENSITIVE

29.1 1.0.4

2022-02-18

- Add incompatibility check, at the moment I'm only aware of TickChanger

29.2 1.0.3

2022-01-30

- Support subcommands (GH #74)
- Support discord.py 2.x
- Support aliases made with the alias cog (GH #75)

29.3 1.0.2

2021-11-26

- Slightly change behaviour

29.4 1.0.1

2021-11-26

- Properly name info command

CHAPTER

THIRTY

CMDLOG

30.1 1.5.2

2022-04-23

- Fix DMs

30.2 1.5.1

2022-04-06

- Fix normal command logging

30.3 1.5.0

2022-04-06

- Remove support for dislash
- Remove support for SlashInjector
- Improve support for dpy slash commands

30.4 1.4.5

2022-03-31

- Add colour formatting to log messages sent to a channel (idea from sraven, **`Discord message<https://discord.com/channels/240154543684321280/931931692619423805/958301149008633876>`_**

30.5 1.4.4

2022-02-28

- Fix logging to channel when the cog is loaded on bot start

30.6 1.4.3

2021-09-05

- Guard dislash.py with TYPE_CHECKING

30.7 1.4.2

2021-09-05

- Add support for dislash.py application commands

30.8 1.4.1

2021-08-28

- Fix AttributeError in sending com log to channel
- Fix AttributeError in handling slash commands from Kowlin's SlashInjector
- Ensure bot has send message permissions when setting log channel
- Fixes CMDLOG-2 and CMDLOG-3 on Sentry

30.9 1.4.0

2021-08-27

- Add new command (`[p]cmdlog channel`) to log commands to a channel

30.10 1.3.1

2021-08-24

- Add opt-in telemetry and error reporting

30.11 1.3.0

2021-08-12

- Support Application Commands (Slash, Message, User), both with slashinjector/dpy 1 and dpy 2

30.12 1.2.1

2021-08-07

- Initial discord.py 2.0 compatibility

30.13 1.3.0

2021-06-23

- Add content logging, by deafult turned off (see command `[p]cmdlog content`)
- Simplify EUD statement
- Add info on how long long since cog load (how long current cache lasts) on log commands

30.14 1.1.0

2021-05-10

- Log command invoke message IDs
- Round cache size to 1 decimal place

30.15 1.0.2

2021-04-22

- Return correct size… I really thought I already did this.

30.16 1.0.1

2021-04-18

- New command to view cache size (`cmdlog cache`)

30.17 1.0.0

2021-04-18

- Initial release

CHAPTER

THIRTYONE

COVIDGRAPH

31.1 1.2.0

2021-11-28

- Add average line

31.2 1.1.1

2021-11-28

- Fix multi work counties not being picked up properly

31.3 1.1.0

2021-11-28

- Support worldwide data, for example `[p]covidgraph cases world`

31.4 1.0.0

2021-11-27

- New cog

CHAPTER

THIRTYTWO

FIVEMSTATUS

32.1 1.0.1

2022-04-14

- Hotfix

32.2 1.0.0

2022-04-14

- New cog

CHAPTER

THIRTYTHREE

GITHUB

Note: This cog is scheduled for deprecation in favour of a new cog *ghissues* which supports buttons, for when they are officially supported in Red

33.1 1.0.1

2021-08-24

- Add opt-in telemetry and error reporting

CHAPTER

THIRTYFOUR

GOOGLETRENDS

34.1 1.1.0

2022-01-12

- Add a URL button to link to Goole Trends, without any extra libs

34.2 1.0.0

2021-11-09

- Initial release

CHAPTER

THIRTYFIVE

MADTRANSLATE

35.1 1.0.3

2022-02-05

- Fix ValueError (#78)

35.2 1.0.2

2021-08-24

- Add opt-in telemetry and error reporting

35.3 1.0.1

2021-06-07

- Add Vex-Cog-Utils stuff

35.4 1.0.0

2021-06-07

- Initial release

CHAPTER

THIRTYSIX

ROLEPLAY

36.1 1.1.0

2022-04-11

- Add custom title & thumbnail for radio embeds
- Don't replace numbers in radio transmission distortion
- Fix radio embed colour

36.2 1.0.1

2022-04-09

- Add deletion after x minutes

36.3 1.0.0

2022-04-09

- New cog

CHAPTER

THIRTYSEVEN

STATTRACK

37.1 1.9.1

2022-03-26

- Fix database file location

37.2 1.9.0

2022-03-26

- Use direct database queries instead of keeping data in memory
- Memory optimisations due to above

37.3 1.8.5

2022-01-30

- Stop using deprecated method `frame.append`

37.4 1.8.4

2022-01-26

- Force 2 writes on load instead of 1

37.5 1.8.3

2022-01-17

- Manually count up unique users to avoid issues with the bot's own cache
- Performance optimisations

37.6 1.8.1

2022-01-13

- Performance optimisations (from my limited testing with 20k users on a relatively weak Windows machine this yields 4-5X faster loops; only 2X on my Ubuntu VPS)

37.7 1.8.0

2022-01-08

- Show min, max, average (, and total where applicable) in the graph embeds, #69
- Use Discord's colours in the plots for user statuses, thanks Epic
- Use rolling averages for messages + command plots
- Make the bot type on export commands

37.8 1.7.1

2021-12-06

- Ensure plot frequency is always 1 or greater, fixing ZeroDivisionError when maxpoints is greater than the actual number of points to plot

37.9 1.7.0

2021-12-05

- New hidden dev commands: `stattrack devimport`, `stattrack debug`
- Significantly improve performance for very large plots (a few months+) by using a maxiumum amount of points to plot, default at 25,000, settable with `stattrack maxpoints`

37.10 1.6.0

2021-12-02

- Allow stats in the same group to be shown on a single graph

37.11 1.5.1

2021-11-28

- Add loop time metric

37.12 1.5.0

2021-11-28

- Add metrics for CPU and Memory usage percentages

37.13 1.4.0

2021-11-09

- Move to plotly for the plotting backend

37.14 1.3.2

2021-09-14

- Fix TypeError in log for when loop overruns

37.15 1.3.1

2021-08-24

- Add opt-in telemetry and error reporting

37.16 1.3.0

2021-08-11

- Move to SQLite driver in Vex-Cog-Utils

37.17 1.1.0

2021-06-25

- Move to SQLite for data storage for superior speed

37.18 1.0.1

2021-06-12

- Count time to save to config seperatleu

37.19 1.0.0

2021-06-02

- Initial release

CHAPTER

THIRTYEIGHT

STATUS

38.1 2.5.4

2022-05-19

- Add missing send

38.2 2.5.3

2022-04-24

- Add Wikimedia's status page

38.3 2.5.2

2022-03-04

- Oops, I broke the cog

38.4 2.5.1

2022-02-19

- No longer pass `user` to bot.embed_requested to remain compatible with Red 3.5 (Red PR #5576)

38.5 2.5.0

2022-02-07

- Add buttons for discord.py 2.0

38.6 2.4.1

2021-09-14

- Limit embed value length in status command, for affected components. This did NOT affect the background loop and automatic sending of updates

38.7 2.4.0

2021-08-26

- Cache status updates, and therefore decrase the cooldown on the *status* command

38.8 2.3.12

2021-08-24

- Add opt-in telemetry and error reporting

38.9 2.3.11

2021-08-16

- Change service base image URL to static.vexcodes.com

38.10 2.3.10

2021-08-07

- Initial discord.py 2.0 compatibility

38.11 2.3.9

2021-06-27

- Improve embed limit handling

38.12 2.3.8

2021-06-22

- Move icons to GH Pages
- Make field name a zero width space for when embed fields are split

38.13 2.3.7

2021-06-17

- Fix edge case KeyError with service restrictions

38.14 2.3.6

2021-06-08

- New service - Fastly
- Handle embed description limits

38.15 2.3.5

2021-05-22

- Update to use Discord's new logo

38.16 2.3.4

2021-05-19

- Fix KeyError which could occur in edge cases

38.17 2.3.3

2021-05-16

- Change the colour for `investigating` to orange (from red)

38.18 2.3.2

2021-05-08

- Dynamic help for avalible services in all commands that previously had them listed

38.19 2.3.0

2021-05-05

- Use dedicated library (`markdownify`) for handling HTML to markdown
- Remove `pytz` for requirements and remove from code.

38.20 2.2.0

2021-05-01

- Use the ABC in the loop and move to VexLoop

38.21 2.1.5

2021-05-01

- Properly handle errors relating to service restrictions when removing a feed
- Improve error handling/logging in update loop
- Limit number of updates sent per service per check to 3 (eg when cog has been unloaded for a while)

38.22 2.1.4

2021-04-23

- Show status of components in command `status`

38.23 2.1.3

2021-04-22

- Use deque for cooldown

38.24 2.1.2

- Handle EUD data deletion requests (return None)

38.25 2.1.1

2021-13-04

- Minor refactoring

38.26 2.1.0

2021-13-04

38.26.1 User-facing changes

- Handle HTML tags for Oracle Cloud

38.26.2 Internal changes

- Utilise an Abstract Base Class
- Add some internal docstrings

38.27 2.0.0, 2.0.1

(backdated)

38.27.1 Important

If the cog fails to load after updating then you'll need to do the following.

Note: If you originally added my repo and didn't name it `vex`, replace `vex` with what you called it throughout.

1. **Uninstall status and remove my repo**

```
cog uninstall status
```

```
repo remove vex
```

2. **Add my repo back and reinstall status**

```
repo add vex https://github.com/Vexed01/Vex-Cogs
```

```
cog install vex status
```

3. **Restart**

```
restart
```

Note: If you haven't configured anything to catch the restart, you'll need to start your bot up again.

You should now be able to load the cog.

38.27.2 User-facing changes

- BREAKING CHANGES: Removed AWS, GCP, Twitter and Status.io. These will be automaticity removed when you update.
- Added the docs page *Status Reference* to see previews for different modes/webhook
- All updates will now included the impact and affected components (see an example at *Status Reference*)
- New service: GeForce NOW (`geforcenow`)

38.27.3 Event Changes for developers

I highly recommend you read the docs page again at the *Status Events* page.

There have been significant changes to both the events.

38.27.4 Internal changes

- Significant re-factoring into more files and folders
- Rewrite of update checking and sending logic
- Implementation of Status API instead of parsing RSS
- Changes to how incidents are stored including config wrapper
- No longer write ETags to config (just cache)

CHAPTER

THIRTYNINE

SYSTEM

39.1 1.3.10

2022-02-07

- Auto-hide loop disks, old behaviour possible with *[p]system disk False*

39.2 1.3.9

2021-08-24

- Add opt-in telemetry and error reporting

39.3 1.3.8

2021-08-11

- Use correct timezone for system uptime

39.4 1.3.7

2021-08-09

- Fix error on d.py 2

39.5 1.3.6

2021-08-07

- Initial discord.py 2.0 compatibility

39.6 1.3.5

2021-06-30

- Change formatting of `system red` and it's corresponding section of `system all`

39.7 1.3.4

2021-06-29

- Fix `system all` non-embed output

39.8 1.3.5

2021-06-27

- Show Red's resource usage in the `system all` command
- Trigger typing for `system red` command
- Use the bot's name for Red's resource usage instead of just "Red"

39.9 1.3.2

2021-06-25

- Correctly display SWAP usage

39.10 1.3.1

2021-06-25

- New command: `[p]system red`

39.11 1.2.7

2021-06-18

- Make the cog compatible with WSL

39.12 1.2.6

2021-06-18

- Use UTC for bot uptime

39.13 1.2.5

2021-06-18

- Handle no CPU frequency data being avalible

39.14 1.2.4

2021-06-13

- Fix formatting of cpu

39.15 1.2.3

2021-06-12

- Add bot uptime to footer

39.16 1.2.2

2021-06-12

- Show uptime in footer for all commands
- Make embed formatting to two columns dynamic

39.17 1.2.1

2021-05-30

- Handle embed limits

39.18 1.2.0

2021-05-30

- Add command `system net`
- Use AsyncIter for the process generator

39.19 1.1.2

2021-05-08

- Dynamic help showing if commands are avablible on your system

39.20 1.1.1

2021-04-09

- Add missing docstring for `system uptime`
- (internal) Add stubs for psutil

39.21 1.1.0

2021-04-08

- **New command: `system uptime`**
  - shows what time the system was booted and how long ago that was
- Internal refactor, splitting commands and psutil parsers into two files

CHAPTER

FORTY

TIMECHANNEL

40.1 1.3.1

2022-01-30

- Show 24 hour time in `tcset short` output
- More useful error message when an incorrect identifier is used

40.2 1.3.0

2022-01-30

- Support 24 hour time by adding `-24h` to a short identifier, for example `[p]tcset create UK: {ni-24h}`

40.3 1.2.2

2021-08-24

- Add opt-in telemetry and error reporting

40.4 1.2.1

2021-08-07

- Initial discord.py 2.0 compatibility

40.5 1.2.0

2021-06-25

- You can now choose your own format. Take a look at `[p]tcset create` for some infomation on how to do so. You'll have to remove old channels with `[p]tcset remove`

40.6 1.1.1

2021-06-07

- Fix inconsistencies

40.7 1.1.0

2021-05-02

- Improve fuzzy timezone search

40.8 1.0.0

2021-05-01

- Initial release

CHAPTER

FORTYONE

UPTIMERESPONDER

41.1 1.0.0

2022-02-09

- “New” cog (moved from bounty repo)
- Cog for responding to uptime monitoring service pings.

CHAPTER

FORTYTWO

WOL

42.1 1.1.0

2022-04-21

- Add suport for setting specific IPs

42.2 1.0.5

2021-08-24

- Add opt-in telemetry and error reporting

42.3 1.0.4

2021-08-20

- More realease testing…

42.4 1.0.3

2021-08-20

- Stil testing release workflow…

42.5 1.0.2

2021-08-20

- Still testing release workflow…

42.6 1.0.1

2021-08-20

- Testing release workflow, please ignore

42.7 1.0.0

2021-05-31

- Initial release

CHAPTER

FORTYTHREE

META DOCS

43.1 2.2.0

2021-06-21

- Directly link to each section at the top of changelog

43.2 2.1.1

2021-04-11

- Change intro at top to link to *Getting my cogs* instead of saying to load the cog
- Bring docs up to date with docstring in all cogs

43.3 2.1.0

2021-04-08

- Start versioning docs
- Fully use changelog

43.4 2.0.0

(backdated)

- Switch to furo theme

CHAPTER

FORTYFOUR

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

B

O