eldonationtracker

Release 5.2.1

Eric Mesa

Sep 05, 2020

CONTENTS:

ELDonation Tracker is both a Python interface to the Extra Life Charity API and a reference implementation, including a GUI, that can be used to provide on-screen donation information and updates when streaming a gaming video or recording a gaming video in OBS or XSplit. For a video explaining how to use the GUI reference implementation, visit: http://djotaku.github.io/ELDonationTracker/ . You may also use the API to build your own applications that access the API. The modules are well documented, see the Module Index below.

Note: Starting 20200227 this project will strictly follow Semantic Versioning as laid out at https://semver.org/

That means:

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

CHAPTER

ONE

INSTALLATION

1.1 Via PyPi

The first thing to decide is whether you want to install ELDonationTracker to your system packages, user packages, or a virtual environment.

1.1.1 Virtual Environment (recommended)

Create the folder you want to work in and cd into it.

```
python3 -m venv .
source ./bin/activate
# when you are done using the program you can type deactivate
# first, make sure you have the latest pip, I've had trouble installing with old␣
↪versions
pip install --upgrade pip
pip install eldonationtracker
# on Windows you may need to type python -m pip install eldonationtracker
# Grab participant.conf from git repo or create based on documentation
# Place participant.conf in persistent location, see the page in documentation
```

1.1.2 System Packages

Note: This is NOT recommended and can cause issues with your system.

```
sudo pip install --upgrade pip
sudo pip install eldonationtracker
# Grab participant.conf from git repo or create based on documentation
# Place participant.conf in persistent location, see the page in documentation
```

1.1.3 User Install

Note: A user install is sometimes a bit buggier than either using a virtual environment or system packages

```
sudo pip install --upgrade pip
sudo pip install eldonationtracker
# Grab participant.conf from git repo or create based on documentation
# Place participant.conf in persistent location, see the page in documentation
```

1.2 Via Github

Here you have a few options depending on what you want to do.

1.2.1 GUI Windows Executable Users

Go to the latest release and download one of the files that ends in "For Windows". You can choose a single executable binary or a zip file. The single executable is simply launched like any Windows application - double-click it. It has a slower startup time than the zip file, but once it's running, there is no performance penalty. For the zip file: unzip it to the location you want to use and then proceed to *Usage*.

New in version 4.4.0: Single-binary executable build added.

Changed in version 4.0.1: Pyinstaller images will no longer be made for Linux as there are issues with the version of libC it links to as well as other side effects from the VM used by the Github CI system. Linux users can still use the GUI via PyPi, Source Code download, or git clone.

1.2.2 Commandline Users and/or Developers

Two options:

1. Go to the latest release and click on "Source Code (zip)" or "Source Code (tar.gz)" depending on whether you're using on Windows or Linux. Then proceed to *Usage*.
2. Go to the main Github page and click on "Clone or Download" and click the button to copy the URL to your clipboard. Then run:

```
git clone https://github.com/djotaku/ELDonationTracker.git
```

And any time you want to get up to the latest version you can just go to that folder and type:

```
git pull
```

The master branch is always equivalent to the latest release (except maybe with more up-to-date documentation) so you should always end up with a working version of ELDonationTracker if you do a git pull. (As long as you're not changing any files. For that reason you may want to move your participant.conf to the persistent location - see *participant.conf* for that location) Then proceed to *Usage*.

CHAPTER

TWO

USAGE

2.1 For All Users

Version 5.0 adds the use of the rich module which uses different font colors on the console output (either the little black window that opens on Windows or the terminal used to launch the program on Linux). The following is the key for the colors used:

- green: something positive for the user. Eg: new donor
- red: something that needs the user's attention and signifies an error.
- magenta: Needs user's attention, but isn't necessarily an error
- bold blue: informative, but does not need the user's attention.

New in version 5.0: Added use of rich module to colorize output

2.2 GUI Windows Executable users (Microsoft Windows Users)

This refers to you if you downloaded a file like "Extra Life Donation Tracker for Windows v5.0.zip" or "eldonation-tracker for Windows v5.0.ext". You may wish to watch a video on how to use the GUI at http://djotaku.github.io/ELDonationTracker/ . But if you prefer to read, continue below.

2.2.1 Launching

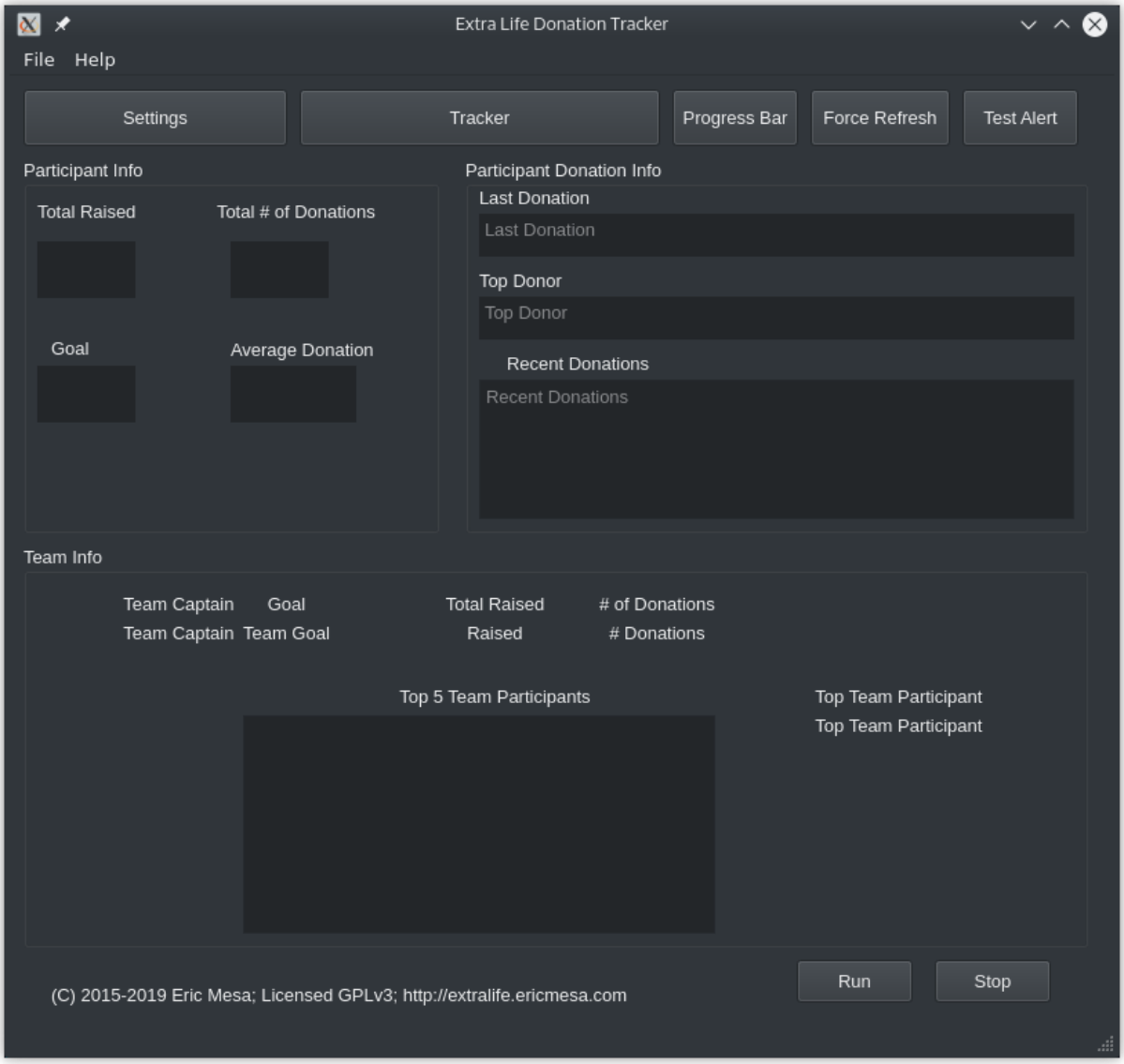
For the zip folder: Go into the folder you unzipped. Find the file called gui.exe and double-click it. If Windows or your anti-virus software throws a warning, click through to allow it to run.

For the single binary: Double-click eldonationtracker.exe to launch it. If Windows or your anti-virus software throws a warning, click through to allow it to run.

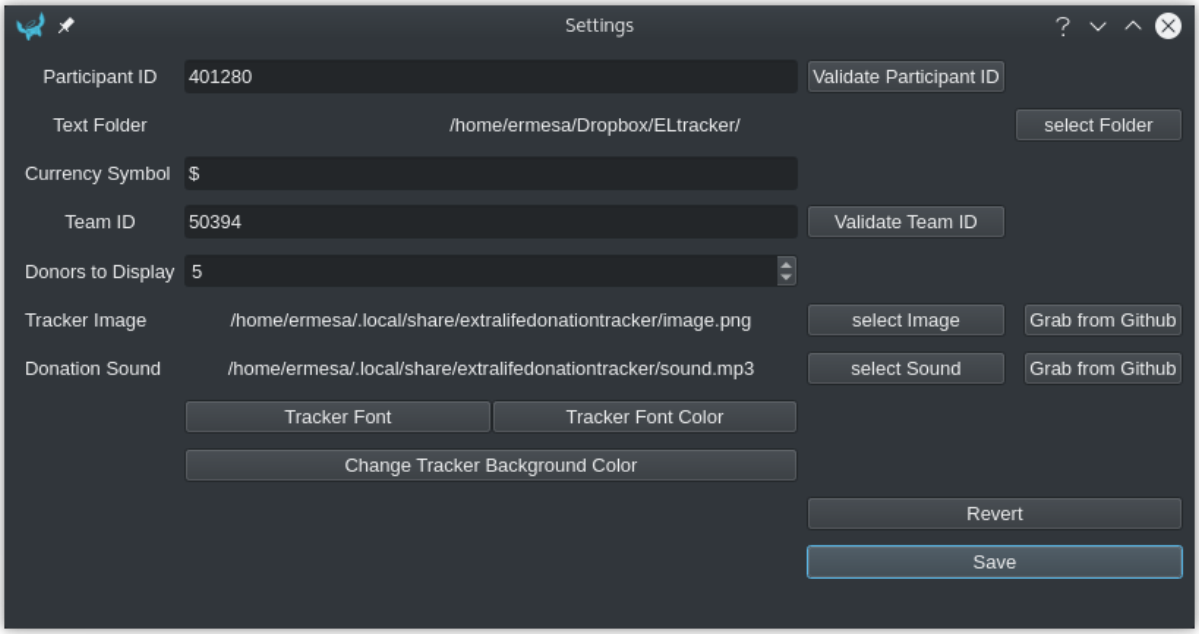
New in version 4.4.0: Single-binary executable.

2.2.2 Using

When you launch the GUI, it will look like this (colors will vary by your OS's color scheme):



If this is not the first time you've used the GUI, it will eventually populate with data (I'll show that at the end). If you've never launched the GUI before and you haven't updated the *participant.conf* during a CLI run, the GUI won't know where to get the data or where to put it once it grabs it from the CLI. So the first thing you want to do is click on the settings window:



The first thing to do here is to enter your participant ID. Where do you find that? Well, it's at the end of your extralife website URL. For example, for 2020, the URL to donate to my campaign is:

https://www.extra-life.org/index.cfm?fuseaction=donorDrive.participant&participantID=401280

As you can see, right at the end it says "participantID=401280". If you look back at that settings window image, you'll see that's what I have in there. If you want to check that you've typed or copied the participant ID correctly, you can click on *Validate Participant ID* and it will attempt to connect to that API endpoint for that participant ID. If it is successful, you have the right participant ID (or coincidentally mistyped someone else's). If it's unsuccessful, the most likely reason is that you mistyped it, but you would also get that outcome if the API is unavailable.

New in version 4.3.0: Validate Participant ID

- The next thing to change is the text folder. This is where eldonationtracker will create text files that you will use as inputs in OBS or XSplit. Every time something changes - you get a donation or the team (if you're part of one) gets a donation, those text files will change (as long as eldonationtracker is running) and so they'll change in real time on your screen in OBS or XSplit. (In OBS or XSplit you set a text source linked to one of those files and it will change as the file contents change)
- Now let's edit the Team ID. If you don't have a team, just make this blank. (No spaces!) Otherwise find your Team ID in a similar way as you found your participant ID. Go to your team page and the ID will be at the end of that URL. If you want to check that you've typed or copied the team ID correctly, you can click on *Validate Team ID* and it will attempt to connect to that API endpoint for that team ID. If it is successful, you have the right team ID (or coincidentally mistyped someone else's). If it's unsuccessful, the most likely reason is that you mistyped it, but you would also get that outcome if the API is unavailable.

New in version 4.3.0: Validate Team ID

- Edit the Donors to Display field. Some of the text files produced by eldonationtracker (lastNDonations, topN-donors, etc) use this number to determine how many donors or donations to write to the file. I usually put it at about five, but I also don't get a lot of donations most years.

The final settings all deal with the tracker window. If you scroll just a little lower in this tutorial you'll see a rectangle with a green background and an image and some text appearing and disappearing. That window is what will appear on your screen when someone donates if you've properly set up OBS or XSplit to capture those windows.

- You should select an image with a transparent background (most likely a gif or png) that will appear when someone donates. I'll show what it'll look like in a minute. If you would like to use the default image, click on *Grab from Github* next to the *select Image* button and it will grab it from GitHub and place it in the XDG location for your system. Make sure to hit *Persist Settings* afterward to save it to your settings.

New in version 4.3.0: The ability to grab the default image from GitHub.

- Also select an MP3 file you would like to play when you get a donation. Keep it shorter than 15 seconds. If you would like to use the default mp3 (my daughter saying "you got a donation!"), click on *Grab from Github* next to the *select Sound* button and it will grab it from GitHub and place it in the XDG location for your system. Make sure to hit *Persist Settings* afterward to save it to your settings.

New in version 4.3.0: The ability to grab the default sound from GitHub.

- With the last 3 buttons you can change the Font type and size, the font color, and the background color. I recommend a size around 48-50 (you may need to type it in yourself if it's not a selectable number). For the background color, it's probably best to stick with the chromakey green I've selected because that makes it incredibly easy for OBS or XSplit to make the background disappear so that on your screen you just see the image and text (not the green background). But if the image you want to use has a lot of green in it, you may need to choose bluescreen blue or some other color that will also work well with OBS or XSplit's chromakey filters.

New in version 4.2.0: Ability to change the font type, size, and color as well as the tracker background color.

Warning: Because of the way QT color chooser dialogue windows work, if you pick a color and hit cancel, it will still change the color in the Tracker window. (whereas you have to click "ok" in the Font chooser window to change the font) If you go back in and pick one of the colors from the palette on the left, you can get it working again. Or you can slide the right-most slider from black to white. Finally, if you can't remember what color you had quitting out of everything without saving should bring back the last color you saved (or the default).

- Finally, it's time to save your settings. Hit *Save*. Your settings will be saved to a special location on your computer so that even as you upgrade (either grab new zip files from Github or update via PyPi or git pull) you won't have to keep inputting your settings. If you have not hit *Save* yet, Revert will reload whatever configuration information was in the file when you hit the *Settings* button.

Changed in version 5.2.0: Changed to just have a Save setting instead of Save and Persistent Settings

OK, now it's time to test that things are working with your settings. Close the settings window and click on *Tracker*. Then hit test alert. If everything was correctly set up in the settings, you should see something like:

And hear the sound you picked. What the text says will depend on whether you've ever run this program before either in GUI or on the commandline. If you've never run it, you'll get a test message. If you have run it and the settings are correctly configured, it should show whatever is in your file called `LastDonationNameAmnt.txt`.

OK, now it's time to hit *Run* and hopefully if all the directions have been followed and I haven't introduced any bugs, it should start grabbing data from the API. You should look at the commandline window for information. Whether you launched the GUI from gui.exe, used PyPi, or python gui, you should have a commandline window showing messages related to what's going on. It should look something roughly like this:

```
Looking for persistent settings at (this path will depend on your system)
Persistent settings found.
Participant.conf version check!
Version is correct
run button
Starting the participant run. But first, reloading config file.
Looking for persistent settings at (this path will depend on your system)
```

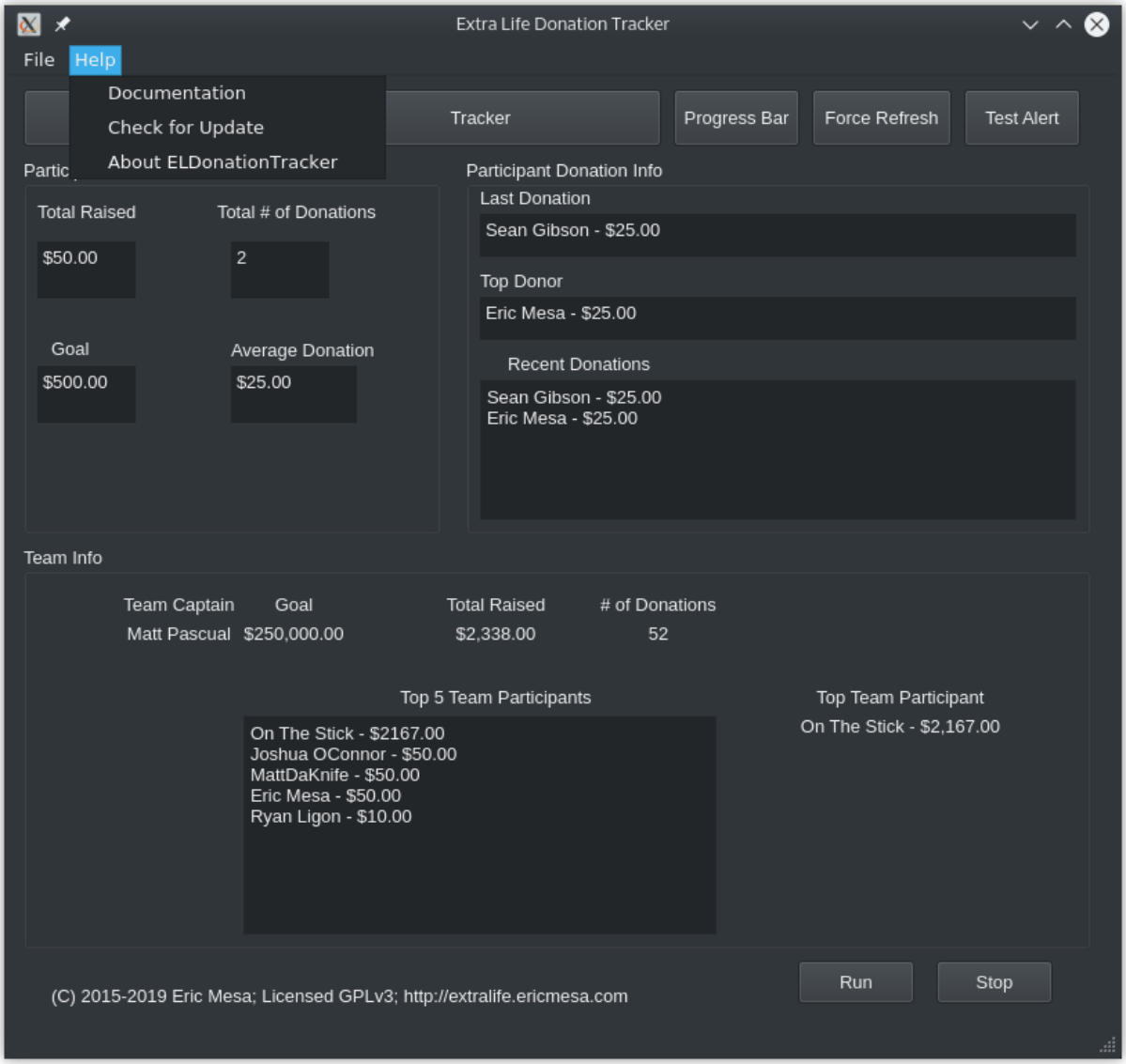
(continues on next page)

(continued from previous page)

```
Persistent settings found.
19:19:10
```

When you're done, be sure to hit stop. When you exit out, it will take a few seconds until it's done before the GUI will disappear. If you Go *File* → *Quit*, that will also trigger it to stop. Again, it'll take a few seconds before it's all cleaned up and ready to disappear from your screen.

Finally, let's quickly go over the help menu items at the top of the GUI.



- *Documentation* will take you to the latest version of this very documentation you're reading now
- *Check for Update* will check if you have the latest version. It will then pop up a window to let you know.
- *About ELDonationTracker* will bring up a window with some URLs and copyright data. Eventually if we start getting more contributors, those would be listed there, too.

2.3 Commandline users (PyPi)

Go to the folder you created in *Installation*. If you don't have the virtual environment activated, start with that:

```
source ./bin/activate
# to check for upgrades
pip install --upgrade eldonationtracker
```

2.3.1 GUI

Make sure you have the *participant.conf* in the persistent location. You can grab the one in the GitHub repo or create your own by looking at the example there. Once the GUI has actually started, you can easily modify the config file via the GUI. To start the GUI:

```
python -m eldonationtracker.gui
```

That should work just fine. Keep an eye on the commandline for any errors or messages from eldonationtracker. The benefit you get from using the GUI is that once the GUI comes up you can click "tracker" to get a window that will display an image and text when a donation is registered. For text instructions on how to use the GUI, go to *Using* or watch the video at http://djotaku.github.io/ELDonationTracker/

eg:

You can also edit the participant.conf settings in a GUI rather than on the commandline and those settings will persist to commandline-only usage.

2.3.2 Commandline Only (No GUI)

Make sure you have the *participant.conf* in the persistent location. You can grab the one in the Github repo or create your own by looking at the example there. To start the commandline only version:

```
python -m eldonationtracker.participant
```

Of course, you can import the modules into your own scripts and modify how you use the code I've written. In that case, you may be interested in the module index to get a good look at the API available to your program.

Changed in version 5.0.0: Command changed from from python -m eldonationtracker.extralifedonations to python -m eldonationtracker.participant

2.4 Commandline users (git)

If you downloaded a zip or tar file, unzip it first, then cd into that directory. If you did a git clone, cd in to that directory. Afterwards, follow along below to create a virtual environment (so as not to mess with your Python installation), grab the required packages, and run the program. (For information on what you should put into participant.conf, see *participant.conf*.

```
python3 -m venv .
source ./bin/activate
# when you are done using the program you can type deactivate
pip install --upgrade pip
```

(continues on next page)

(continued from previous page)

```
pip install -r requirements.txt
# on Windows you may need to type python -m pip install -r requirements.txt
# edit participant.conf
cd eldonationtracker
# for the GUI:
python gui.py
# for the commandline only
python participant.py
```

The benefit you get from using the GUI is that once the GUI comes up you can click "tracker" to get a window that will display an image and text when a donation is registered. For text instructions on how to use the GUI, go to *Using* or watch the video at http://djotaku.github.io/ELDonationTracker/

eg:

You can also edit the settings in a GUI rather than on the commandline. Once the settings are configured, hit the run button. You should get the same output on the commandline as you would if you weren't running the GUI. Check there for any errors or messages from the program.

CHAPTER

THREE

PARTICIPANT.CONF

This is the configuration file for the program.

3.1 Example

Here's an example from my config file in 2020:

```
{
"Version" : "1.0",

"extralife_id" : "401280",

"text_folder" : "/home/ermesa/Dropbox/ELtracker/",

"currency_symbol" : "$",

"team_id" : "50394",

"tracker_image": "Engineer.png",

"donation_sound": "/home/ermesa/Programming Projects/python/extralife/Donation.mp3",

"donors_to_display": "5"

"font_family": "Liberation Sans",

"font_size": 52,

"font_italic": true,

"font_bold": 75,

"font_color": [255, 255, 255, 255],

"tracker_background_color": [0, 255, 0, 255]

}
```

If you are using the GUI, you can edit the file with the GUI and it should always turn out OK. If you are editing it in a text editor, the important thing to remember is the quotation marks.

3.2 Locations

The program will look in 3 locations.

First, it will look in the XDG official locations. If you use "save persistent settings" in the GUI, it will save to this location. On Linux this will be in $HOME/.config/extralifedonationtracker. On Windows it will be C:\Users\username\AppData\Roaming\config\extralifedonationtracker where you replace username with your Windows username.

Second, it will look up two directory levels. If you cloned from Github or grabbed a Source Code.zip or .tar.gz, this would be in the first directory you cd into - the one that has requirements.txt, README.md, etc.

Third, it will look in the directory where it is being run. If you are using the GUI single executable, this means the same folder as gui.exe (on Windows) or gui (on Linux).

As long as it's in one of those directories, it will be fine. Otherwise it will attempt to grab a copy from Github.

3.3 Config Values

- Version: this should be left alone. It's to know if a persistent config needs updating
- extralife_id: this should be your extralife id, the end of your campaign URL
- text_folder: this is where the text files generated by the program will be placed
- currency_symbol: the prefix for your money
- team_id: if you're on a team, grab from team page URL. If not, put null without quotation marks
- tracker_image: if you're using the GUI, what you want to appear when you get a donation
- donation_sound: if you're using GUI, what you want to hear when there's a donation
- donors_to_display: controls how many donors/donations appear in files that have more than one

The values corresponding to the tracker (everything after donors_to_display) is best configured in the GUI to ensure you're getting values that make sense on your system.

CHAPTER

FOUR

CALL_ABOUT

Contains programming logic for the about window in the GUI.

eldonationtracker.call_about.**main**()

CHAPTER

FIVE

CALL_SETTINGS

Contains the programming logic for the settings window in the GUI.

eldonationtracker.call_settings.**main**(*participant_conf*)
Launch the window.

CHAPTER

SIX

CALL_TRACKER

A window that displays the last donation. Useful during streaming.

eldonationtracker.call_tracker.**main**(*participant_conf*)
Launch the window.

CHAPTER

SEVEN

DONATION

A class to hold the Donation attributes and methods.

class eldonationtracker.donation.**Donation**(*name*, *message*, *amount*, *donor_id*, *avatar_url*, *donation_date*, *donation_id*)

Bases: object

Donation Attributes.

Class exists to provide attributes for a donation based on what comes in from the JSON so that it doesn't have to be traversed each time a donor action needs to be taken.

Parameters

- **name** (*str*) – the name of the donor for this donation. If the donor wished to stay anonymous, the variable is set to/ "Anonymous"
- **message** (*str*) – the message associated with the donation.
- **amount** (*int*) – the amount of the donation. If they blocked it from showing it is set to 0.

get_amount()

eldonationtracker.donation.**format_donation_information_for_output**(*donation_list: list*, *currency_symbol: str*, *donors_to_display: str*, *team: bool*) → dict

Format the donation attributes for the output files.

eldonationtracker.donation.**get_donations**(*donations: list*, *donation_url: str*) → list

Get the donations from the JSON and create the donation objects.

If the API can't be reached, the same list is returned. Only new donations are added to the list at the end.

Parameters

- **donations** – A list consisting of donor.Donation objects.
- **donation_url** – The URL to go to for donations.

Returns A list of donor.Donation objects.

CHAPTER

EIGHT

DONOR

A donor.

class eldonationtracker.donor.**Donor**(*json*)

Bases: object

Donor Attributes.

Class exists to provide attributes for a donor based on what comes in from the JSON so that it doesn't have to be traversed each time a donor action needs to be taken.

Parameters

- **json** (*json*) – JSON attributes from the API
- **self.name** – donor's name if provided, else Anonymous
- **self.donor_id** – the ID assigned by the API (currently not used)
- **self.image_url** – the URL for the donor's avatar (currently not used)
- **self.amount** – the sum of all donations the donor has made this campaign
- **self.number_of_donations** – the number of donations the donor has made this campaign

json_to_attributes(*json*)

Convert API JSON values to Donor attributes.

May be overwritten by child classes.

Parameters **json** (*json*) – JSON attributes from the API

CHAPTER

NINE

PARTICIPANT

Grabs Participant JSON data and outputs to files.

class eldonationtracker.participant.**Participant**(*config*)

Bases: object

Owns all the attributes under the participant API.

Also owns the results of any calculated data.

Donor Drive API api info at https://github.com/DonorDrive/PublicAPI

Parameters

- **self.extralife_id** (*int*) – the participant's extra life ID
- **self.text_folder** (*str*) – where the output text files will be written on disk
- **self.currency_symbol** (*str*) – for the output text files
- **self.donors_to_display** (*int*) – for text files that display multiple donors (or donations), the number of them that should be written to the text file.
- **self.participant_url** (*str*) – API info for participant
- **self.donation_url** (*str*) – donation API info
- **self.participant_donor_url** (*str*) – API info for donors. Useful for calculating top donor.
- **self.total_raised** (*int*) – total amount raised by the participant
- **self.number_of_donations** (*int*) – the number of donations received by the participant
- **self.average_donation** (*int*) – The average amount of donations for this participant
- **self.goal** (*int*) – The goal for the amount of the money the participant wishes to raise
- **self.participant_formatted_output** (*dict*) – a dictionary holding data about the participant
- **self.my_team** (*cls: eldonationtracker.team*) – An instantiation of a team class for the participant's team.
- **self.donation_list** (*list*) – a list of Donation class objects made of donations to this participant
- **self.donation_formatted_output** (*dict*) – a dictionary holding values for txt output

output_donation_data() → None
Write out text files for donation data.

If there have been donations, format the data (eg horizontally, vertically, etc) and output to text files. If there have not yet been donations, write default data to the files.

output_donor_data() → None
Write out text files for donor data.

If there have been donations, format the data (eg horizontally, vertically, etc) and output to text files. If there have not yet been donations, write default data to the files.

output_participant_data() → None
Format participant data and write to text files for use by OBS or XSplit.

A public method to do the above. Also called from the main loop.

run() → None
Run to get participant, donation, donor, and team data and output to text files.

set_config_values() → None
Set participant values, create URLs, and create Team.

update_donation_data() → None
Update donation data.

As of 5.0 it just updates the list of donations. There may be more donation-related updating in future versions.

update_donor_data() → None
Update donor data.

As of 5.0 it only grabs the top donor. There may be more donor-related updates in the future.

update_participant_attributes() → None
Update participant attributes.

A public method that will update the Participant object with data from self.participant_url.

Also called from the main loop.

write_text_files(*dictionary: dict*) → None
Write OBS/XSplit display info to text files.

It uses the helper function extralife_IO.write_text_files to handle the task.

Parameters **dictionary** (*dict*) – Dictionary containing values to write to text files . The key will become the filename. The value will be written to the file.

CHAPTER

TEN

EXTRALIFE_IO

Holds all the file and internet input and output.

class eldonationtracker.extralife_io.**ParticipantConf**
Bases: object

Holds Participant Configuration info.

Parameters

- **cls.participant_conf_version** – version of participant.conf
- **cls.version_mismatch** – Initialized to False.If true, the user has a different version of the participant.conf.
- **cls.fields** – A dictionary initialed to None for all fields.
- **self.xdg** – By using the PedanticPackage, the directory will be created if it doesn't already exist.
- **self.participantconf** – holds a dictionary of the user's config file.

fields: dict = {'currency_symbol': None, 'donation_sound': None, 'donors_to_display

get_cli_values() → Tuple[Any, Any, Any, Any, Any]
Return data required for a CLI-only run.

Returns A tuple of strings with config values needed if only running on the commandline.

get_font_info()
Return values needed to change the font for the tracker.

Returns A tuple of strings and a list for the color.

get_github_config()

get_gui_values() → Tuple[Any, Any, Any, Any, Any, Any, Any, Any, Any, Any, Any, Any, Any]
Return values needed for the GUI.

Returns A tuple of strings with config values needed if only running the GUI. The two background colors are lists.

get_if_in_team() → bool
Return True if participant is in a team.

Returns True if the participant is in a team.

get_text_folder_only() → str
Return text folder data.

Returns A string with the text folder location.

get_tracker_assets(*asset: str*)

get_tracker_background_color()
Return value needed to change the tracker background color

Returns A list representing the RGB value for the background

get_tracker_image() → str
Return the tracker image location on disk.

Returns Location of tracker image on disk.

get_tracker_sound() → str
Return the donation sound image location on disk.

Returns location of the donation sound on disk.

get_version()
Return version.

get_version_mismatch() → bool
Return bool of whether there is a version mismatch.

Returns True if the version the user is running is not the same as what this program has as its version.

load_json() → dict
Load in the config file.

Checks in a variety of locations for the participant.conf file. First, it checks in the persistent settings location (XDG_CONFIG_HOME) . Then it checks the current directory and one level up. Otherwise it raises an exception and the program stops.

Returns A dictionary representing the JSON config file.

Raises FileNotFoundError

participant_conf_version: str = '2.0'

reload_json()
Reload JSON and update the fields.

update_fields()
Update fields variable with data from JSON.

version_mismatch: bool = False

write_config(*config: dict*, *default: bool*)
Write config to file.

Only called from GUI. Commandline user is expected to edit participant.conf manually. Afterward it triggers self.reloadJSON to have the program run with the updated config.

Parameters

- **config** – A dictionary holding the config values.
- **default** – If True, will save the file in the current directory.

eldonationtracker.extralife_io.**get_json**(*url: str*, *order_by_donations: bool = False*) → dict
Grab JSON from server.

Connects to server and grabs JSON data from the specified URL. The API server should return JSON with the donation data.

Parameters

- **url** – API URL for the specific json API point.
- **order_by_donations** – If true, the url param has data appended that will cause the API to return the data in descending order of the sum of donations.

Returns JSON as dictionary with API data.

Raises HTTPError, URLError

eldonationtracker.extralife_io.**multiple_format**(*donors*, *message: bool*, *horizontal: bool*, *currency_symbol: str*, *how_many: int*) → str

Format string for output to text file.

This function is for when there is are multiple donors or donations. For example, for the text file holding the Top 5 donors.

Parameters

- **donors** – An iterable of Donors or Donations class objects.
- **message** – If True, the message (if any) from the donor object :param horizontal: If True, format the message horizontally. Else vertically.
- **currency_symbol** – The currency symbol to append to the return string.
- **how_many** – A number for how many donors/donations to append to the string.

Returns A string containing the inputs formatted.

Horizontal example:

John Doe - $100.00 - Thanks for raising money for the kids. | Jane Doe - $25.00 - Hurray!

Vertical example:

John Doe - $200.00 - a message

Jane Doe - $75.00 - another message

eldonationtracker.extralife_io.**single_format**(*donor*, *message: bool*, *currency_symbol: str*) → str

Format string for output to text file.

This function is for when there is only one donor or donation. For example, for the text file holding the most recent donation.

Parameters

- **donor** – Donor or Donation class object.
- **message** – If True, the message (if any) from the donor object :param currency_symbol: The currency symbol to append to the return string.

Returns A string containing the inputs formatted. For example:

John Doe - $100.00 - Thanks for raising money for the kids.

eldonationtracker.extralife_io.**validate_url**(*url: str*)

eldonationtracker.extralife_io.**write_text_files**(*dictionary: dict*, *text_folder: str*)
Write info to text files.

The dictionary key will become the filename and the values will be the content of the files.

Parameters

- **dictionary** – The dictionary with items to output.

- **text_folder** – The directory to write the text files.

CHAPTER

ELEVEN

GUI

The main GUI window.

eldonationtracker.gui.**main**()

CHAPTER

TWELVE

IPC

Writes to an IPC value to allow different modules to pass information.

This is used to allow the tracker to know that new data needs to be read in because a new donation has occurred.

eldonationtracker.ipc.**write_ipc**(*folder: str*, *value: str*)

Write to the IPC file.

This is used to let the call_tracker module know that a new donation has been made.

Parameters

- **folder** – The location of trackerIPC.txt
- **value** – The value to write to the file.

Raises IOError

CHAPTER

THIRTEEN

TEAM

Contains classes pertaining to teams.

class eldonationtracker.team.**Team**(*team_id: str*, *output_folder: str*, *currency_symbol: str*, *donors_to_display: str*)

Bases: object

Hold Team API Data.

Parameters

- **self.team_id** – The team's ID in the API
- **self.team_url** – URL to the team JSON API
- **self.team_participant_url** – URL to the JSON api for participants in the team.
- **self.team_donation_url** – URL to the JSON api for donations to the team
- **self.output_folder** – The folder for the output text files
- **currency_symbol** – for formatting text
- **self.team_info** – a dictionary to values for output to text files
- **self.participant_calculation_dict** – dictionary holding output for txt files
- **self.top_5_participant_list** – a list of the top 5 team participants by amount donated.
- **self.team_json** – A dictionary to hold JSON info from the API
- **self.team_goal** – the fundraising goal of the team.
- **self.team_captain** – The name of the team captain.
- **self.total_raised** – The total amount raised by the team.
- **self.num_donations** – The total amount of donations to the team.
- **self.top_5_participant_list** – The top 5 participants in the team
- **self.participant_list** – a list of the most recent participants

donation_run() → None
Get and calculate donation information.

participant_run() → None
Get and calculate team participant info.

team_api_info() → None
Get team info from API.

team_run() → None
A public method to update and output team and team participant info.

write_text_files(*dictionary: dict*) → None
Write info to text files.

Parameters **dictionary** – The dictionary containing the values to write out to text files.

CHAPTER

FOURTEEN

TEAM_PARTICIPANT

class eldonationtracker.team_participant.**TeamParticipant**(*json*)

Bases: *eldonationtracker.donor.Donor*

Participant Attributes.

Inherits from the donor class, but over-rides the json_to_attributes function.

API variables:

Parameters

- **self.name** – participant's name or Anonymous
- **self.amount** – the sum of all donations by this participant
- **self.number_of_donations** – number of all donations by this participant
- **self.image_url** – the url of the participant's avatar image (not used)

json_to_attributes(*json*)

Convert JSON to Team Participant attributes.

Parameters **json** – JSON attributes from API

CHAPTER

FIFTEEN

UPDATE_AVAILABLE

Return true if there is an update available.

An update is available if the version on PyPi is higher than the version the user has on their machine.

eldonationtracker.utils.update_available.**get_pypi_version**(*url: str*) → str
Use PyPi JSON API to get latest version.

Returns A string with the version number.

eldonationtracker.utils.update_available.**main**() → bool
Get the latest version on PyPi and compare to current version.

Made the decision to return false if the URL couldn't be reached rather than make the user look for an update that might not be there.

Returns True if there's an update available. False if up to date.

eldonationtracker.utils.update_available.**update_available**(*pypi_version: str*, *current_version: str*) → bool
Use semver module to calculate whether there is a newer version on PyPi.

Returns True if the PyPi version is higher than the version being run. Returns false if the version being compared to PyPi is equal or greater than the PyPi version.

CHAPTER

SIXTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

J

L

M

O

P

R

S

T

U

V

W