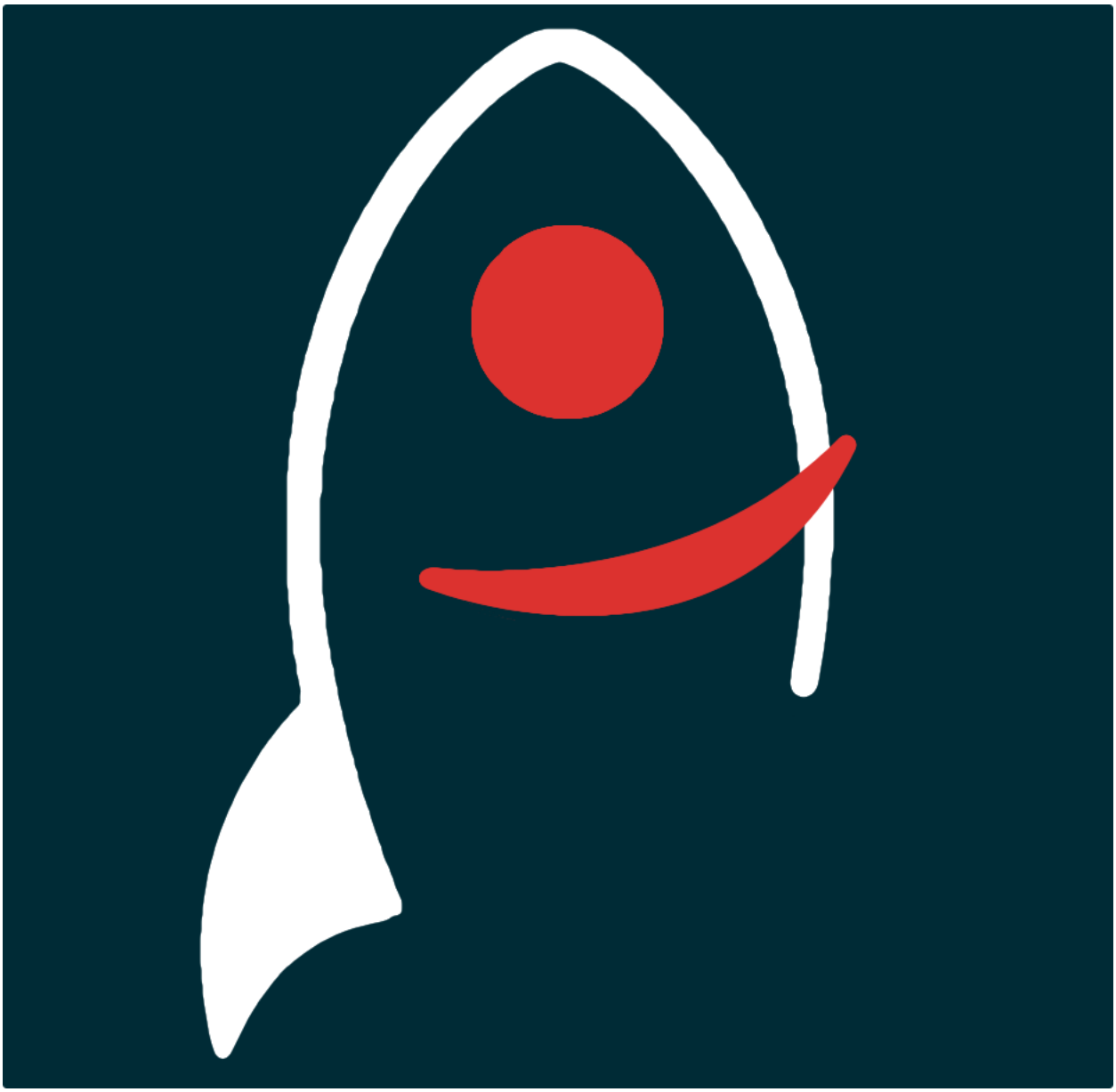


gocart Documentation

Release v0.4.8

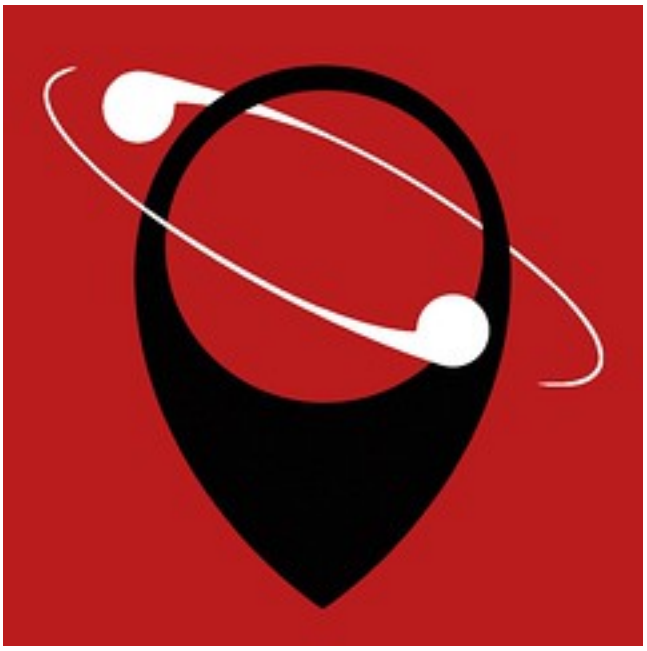
Dave Young

2023

CONTENTS

DOI 10.5281/zenodo.7979470

gocart is a python package and command-line suite used to consume GCN Kafka streams.



Documentation for gocart is hosted by Read the Docs (development version and main version). The code lives on github. Please report any issues you find here.

CHAPTER

ONE

FEATURES

- Listen to GCN Kafka streams (currently LIGO-Virgo-KAGRA only) and write alerts and skymaps to the local filesystem.
- Ability to 'echo' the Kafka stream by re-listening to the last N days of alerts (provided the alerts are still hosted on the GCN Kafka cluster).
- Alert content, FITS Header data and some extra value-added event parameters are written to a single machine-readable YAML file.
- The healpix skymaps are optionally converted to ascii files (one row per healpix pixel) and/or rendered as aitoff plots.
- Create your own plugin scripts to run whenever an alert is parsed.
- works well in conjunction with skyTag.

CHAPTER

TWO

INSTALLATION

The easiest way to install gocart is to use `conda`:

```
conda create -n gocart python=3.9 pip gocart -c conda-forge
conda activate gocart
```

To upgrade to the latest version of gocart use the command:

```
conda upgrade gocart -c conda-forge
```

It is also possible to install via pip if required:

```
pip install gocart
```

Or you can clone the github repo and install from a local version of the code:

```
git clone git@github.com:thespacedoctor/gocart.git
cd gocart
python setup.py install
```

To check installation was successful run `gocart -v`. This should return the version number of the install.

CHAPTER

THREE

INITIALISING GOCART

Before using gocart you need to use the `init` command to generate a user settings file. Running the following creates a yaml settings file in your home folder under `~/.config/gocart/gocart.yaml`:

```
gocart init
```

The file is initially populated with gocart's default settings which can be adjusted to your preference.

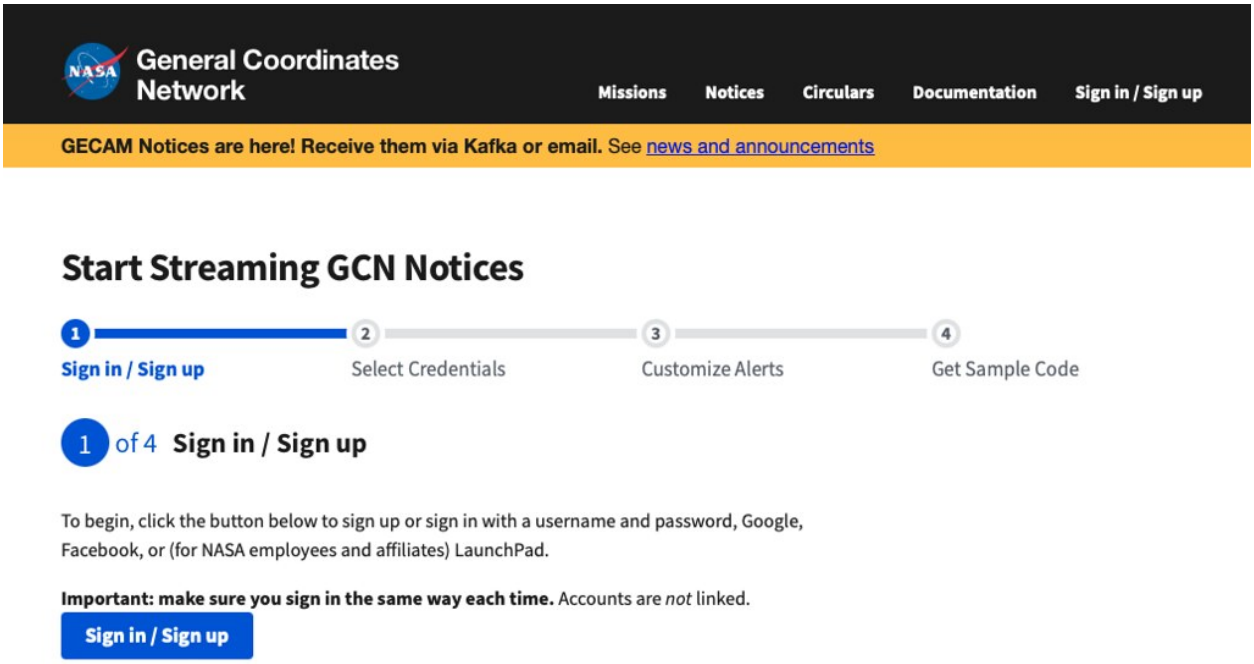
If at any point the user settings file becomes corrupted or you just want to start afresh, simply trash the `gocart.yaml` file and rerun `gocart init`.

CHAPTER

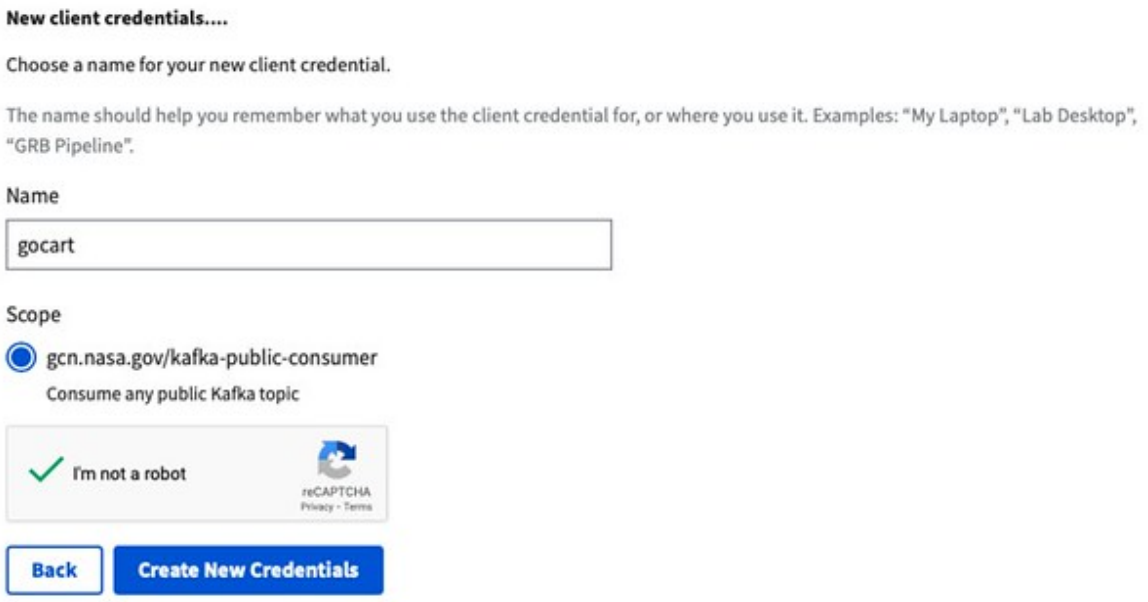
FOUR

GCN KAFKA CREDENTIALS

When you first come to use gocart, you will need to add a GCN Kafka Client ID and Secret to the `gocart.yaml` settings file. To get these credentials you will need to signup for a GCN Notices account here.



Once signed in, fill in a suitable name for your new client (any name will do, so gocart is as good a name as any). Fill in the captcha and click 'Create New Credentials'.



In the 'Customize Alerts' section, don't be concerned about what Notice Types to select as these are only used to generate template code in the next section. Not selecting any is fine. Click the 'Generate Code' button. You will now be presented with your newly created client credentials within the body of the code snippet:

```
# Connect as a consumer.
# Warning: don't share the client secret with others.
consumer = Consumer(client_id='24658srhmdd0a0vu2q2sfdace1',
                    client_secret='1cn0dfhp3nqfd4osglbi2g3fla0is1ael66fbvapv9jfdcircuqc')
```

Open your gocart settings file at `~/.config/gocart/gocart.yaml` and copy and paste these credentials into the appropriate place.

```
gcn-kafka:
    client_id: 24658srhmdd0a0vu2q2sfdace1
    client_secret: 1cn0dfhp3nqfd4osglbi2g3fla0is1ael66fbvapv9jfdcircuqc
    group_id: 205426622397915272571877153567276011810
```

Don't worry about the `group_id` parameter, it's initially set to XXXX but gocart will replace this with a unique value when it's first run. It is this `group_id` that allows GCN Kafka to remember which alerts you have or have not heard before. Note the example client above has been deleted, so the credentials quoted here will not work.

You are now ready to start using gocart.

CHAPTER

FIVE

SKYTAG

gocart works very well in conjunction with skyTag, a tool to 'annotate' transient sources or galaxies with the percentage credibility region they reside within on a given HealPix sky map.

CHAPTER

SIX

HOW TO CITE GOCART

If you use `gocart` in your work, please cite using the following BibTeX entry:

```
@software{Young_gocart,
author = {Young, David R.},
doi = {10.5281/zenodo.7970743},
license = {GPL-3.0-only},
title = ,
url = {https://github.com/thespacedoctor/gocart}
}
```

6.1 Listening to and Echoing an Alert Stream

Once your credentials have been added to the `gocart.yaml` file, you are ready to begin listening to a GCN Kafka stream (LVK only so far). To do so, activate the gocart conda environment and run the command:

```
gocart listen
```

That's it. The listener will run in daemon mode (running as a background task) and download new events and alerts as they are added to the Kafka stream in real-time. If you are parsing the test/mock events (see the `parse_mock_events` setting), you will collect a set of event alerts every ~1hr.

You can check on the status of the listener with `gocart status`. This will tell you if the listener is running or not.

To stop the listener run `gocart quit`, or to restart run `gocart restart`.

If you stop the listener or it falls over for some reason, once you reconnect it will download any event-alerts you missed in the interim.

To relisten the event alerts from the last few days you can run the `echo` command. To request the last 3 days of alerts run the command:

```
gocart echo 3
```

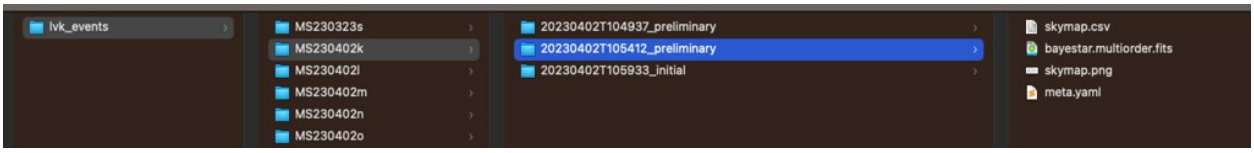
Note, alerts only remain on the GCN-Kafka stream for a finite period of time and you may not be able to relisten to alerts older than a week or so.

6.2 Event and Alert Directories and Files Explained

The location of where event-alert maps are written to file is set via the `download_dir` setting in the settings file. Under this top-level directory, one directory is created per LVK superevent; the name of the directory is the name of the event. Under each event directory, there is one directory for each alert sent for that event. Alerts come in one of 5 types:

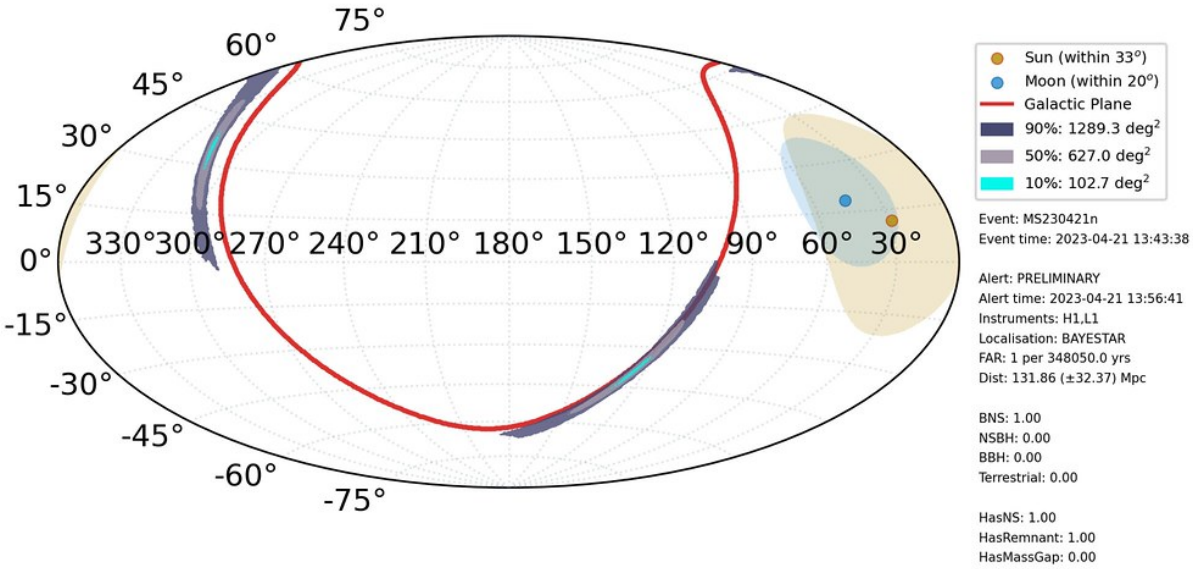
- Early Warning
- Preliminary
- Initial
- Update
- Retraction

The alert folders are named `<alertTimeStamp>_<alertType>`, where the `alertTimeStamp` is the time the alert was issued (not the time of the actual event).



The alert folder host various files:

1. The multi-order healpix skymap issued with the alert (e.g. `bayestar.multiorder.fits`)
2. `meta.yaml` a metadata file containing the contents of the actual alert, combined with info data from the map FITS header and some extra value-added content such as map sky-areas etc.
3. `skymap.csv` is an ascii representation of a single order healpix skymap, with one row per pixel and giving sky-coordinates, probability and distance for each pixel.
4. `skymap.png` is a aitoff rendering of the skymap, galactic plane, sun and moon position and some extra information useful for planning observations.



6.3 Settings

There are many options to configure to your liking within the settings file, and all settings are documented within the comments in the settings file. For example here are the LVK stream settings:

```
lvk:
    # DOWNLOAD MOCK EVENT ALERTS? USEFUL AS A HEARTBEAT MONITOR
    parse_mock_events: True
    # DOWNLOAD PRODUCTION EVENT ALERTS?
    parse_real_events: False
    # LOCATION TO DOWNLOAD EVENT ALERT AND MAPS TO
    download_dir: ~/lvk_events
    # CONVERT MAPS TO AITOFF PLOTS?
    aitoff:
        convert: True
        day_night: False
        sun_moon: False
        galactic_plane: True
    # CONVERT MAP TO ASCII FILE - ONE ROW PER HEALPIX PIXEL
    ascii_map:
        convert: True
        # THE SIZE OF HEALPIX PIXELS TO RESOLVE SKY TO. NSIDE = 64 IS ~0.84 deg2 PER␣
↪PIXEL.
        nside: 64
    # WRITE ORIGINAL JSON ALERTS TO FILE?
    json: False

    # UP FRONT FILTERING OF ALERTS. ONLY IF AN ALERT PASSES 1 OR MORE OF THESE␣
↪FILTERS WILL THE ALERT (AND ASSOCIATED ASSETS) GET WRITTEN TO FILE
    # AN ALERT MUST PASS ALL INDIVIDUAL CRITERIA WITHIN A FILTER TO PASS
    filters:
        - name: significant bursts
          alert_types: [initial, update]
          burst: True
          significant: True
        - name: general
          alert_types: [initial, update, retraction]
          ns_lower: 0.9
          far_upper: 1.6e-08
          dist_upper: 500
          area90_upper: 2000
        - name: high significance
          alert_types: [initial, update, retraction]
          ns_lower: 0.99
          far_upper: 1.6e-10
          dist_upper: 250
          hasns_lower: 0.9
          hasremnant_lower: 0.5
        - name: live event
          event_dir_exist: True
```

6.3.1 Alert Filtering

Within the settings file, you have the option to write 'filters'. An alert coming from the alert stream must pass one or more of the filters before its metadata and maps are written to file. Each filter must be given a descriptive name. Within the LVK module the filtering criteria available are as follows:

- `alert_types`: the alert type must be found within this list. Alerts types are *early_warning, preliminary, initial, update* and *retraction*.
- `ns_lower`: the lower limit of the `NS+BHNS` found in the event classification.
- `far_upper`: an upper limit for the False Alarm Rate
- `dist_upper`: an upper limit for `DISTMEAN` found in the alert map FITS header (Mpc).
- `area90_upper`: an upper limit to the sky-area containing the 90% credibility region of the event (square degrees).
- `hasns_lower`: a lower limit for the `HasNS` property.
- `hasremnant_lower`: a lower limit for the `HasRemnant` property.
- `event_dir_exist`: the event directory already exists, i.e. a previous alert passed the filtering criteria.
- `burst`: the alert results from an unmodelled burst event.
- `significant`: is the significance flag set to True or False

When running gocart in listen or echo mode, the status of the alert's filter pass or fail state will be reported.

```
----------------------------------------
EVENT: MS230426l detected at 2023-04-26T11:38:32.211 UTC (CBC)
ALERT: INITIAL reported at 2023-04-26T12:02:13 UTC (+23.68 mins)
FAR: 1 per 348050.0 yrs
CLASSIFICATION: {'BNS': 1.0, 'NSBH': 0.0, 'BBH': 0.0, 'Terrestrial': 0.0}
PROPERTIES: {'HasNS': 1.0, 'HasRemnant': 1.0, 'HasMassGap': 0.0})
The alert passes the general filter
The alert passes the high significance filter
----------------------------------------



----------------------------------------
EVENT: MS230426m detected at 2023-04-26T12:49:40.577 UTC (CBC)
ALERT: PRELIMINARY reported at 2023-04-26T12:52:17 UTC (+2.61 mins)
FAR: 1 per 348050.0 yrs
CLASSIFICATION: {'BNS': 1.0, 'NSBH': 0.0, 'BBH': 0.0, 'Terrestrial': 0.0}
PROPERTIES: {'HasNS': 1.0, 'HasRemnant': 1.0, 'HasMassGap': 0.0})
The alert fails the general filter. Alert type is preliminary.
The alert fails the high significance filter. Alert type is preliminary.
----------------------------------------
```

6.4 Gocart Plugins

gocart has the ability to run 'plugins'. These are Python scripts gocart can trigger every time a message is read from the GCN Kafka stream. To run gocart *with* plugins, pass the `--plugins, -p` flag to any of the commands. For example:

```
gocart -p listen
```

or

```
gocart -p echo 0.3
```

In `~/.config/gocart/plugins` you will find a template plugin script called `gp_template.py` which can be run as a stand-alone script like so:

```
python ~/.config/gocart/plugins/gp_template.py <pathToAlertDirectory>
```

For example, if you run:

```
python ~/.config/gocart/plugins/gp_template.py ~/lvk_events/mockevents/MS230423x/
↪20230424T000204_initial/
```

the template script prints the following to the command line:

```
Here are the alert files generated from INITIAL alert of event MS230423x:
bayestar.multiorder.fits
skymap.csv
meta.yaml
skymap.png
MS230423x-initial.json
```

6.4.1 Create Your Own Plugin

The quickest way to create your own plugin is to copy the template script and modify it:

```
cd ~/.config/gocart/plugins/
cp gp_template.py gp_sound_the_alarm.py
```

Open the new script in your favourite text editor and edit the code within the `plugin` function:

```
def plugin(
        log,
        settings,
        alertFiles,
        alertMeta):
    """*this is the gocart plugin function that will be run when an alert is read*

    **Key Arguments:**

    - ``log`` -- logger
    - ``settings`` -- these are the gocart settings, you can add extra settings to␣
↪the gocart.yaml settings file and they will be read here.
    - ``alertFiles`` -- a list of all the files generated by gocart for the alert
    - ``alertMeta`` -- a dictionary of the alert metadata from the json alert, FITS␣
↪Header and gocart generated extras.
```

(continues on next page)

(continued from previous page)

```
    """
    log.debug('starting the ``plugin`` function')

    alertType = alertMeta["ALERT"]["alert_type"]
    evertId = alertMeta["ALERT"]["superevent_id"]

    print(f"Here are the alert files generated from {alertType} alert of event
↪{evertId}:")
    for f in alertFiles:
        print(f)

    log.debug('completed the ``plugin`` function')
    return None
```

Before you start developing the script, make sure to generate a few alert directories to test on with (with `parse_mock_events` setting set to True):

```
gocart echo 1
```

Now you can test your script with something similar to this (just make sure you are pointing to an alert directory that exists):

```
python ~/.config/gocart/plugins/gp_sound_the_alarm.py ~/lvk_events/mockevents/
↪MS230423x/20230424T000204_initial/
```

Happy hacking!

6.5 Release Notes

v0.4.8 - September 4, 2023

- **ENHANCEMENT**: added a 'significant' filter option to filter on significant = True or False
- **FIXED**: bug in burst filtering

v0.4.7 - August 8, 2023

- **FIXED**: fixing parsing issue for initial burst alerts containing image links as property and classification parameters

v0.4.6 - June 8, 2023

- **REFACTOR**: making plots consistent across platforms by including a MPL style file
- **FIXED**: area of contours now included in plots if generated from a plugin

v0.4.5 - June 6, 2023

- **FEATURE**: added ability to plot square footprints on to the aitoff skymaps
- **ENHANCEMENT**: significance explicitly added to RHS sidebar of maps
- **REFACTOR**: colours of map contours adjusted to pastel colours
- **REFACTOR**: transparency of sun, moon, galactic plane adjusted for greater clarity
- **FIXED**: contours often getting tangled at the poles of the aitoff maps and rendering incorrectly. Solved by 'chunking' the filled contours with matplotlib

v0.4.4 - June 1, 2023

- **REFACTOR**: `get_moon` has being depreciated in astropy, moving to get_body("moon")
- **FIXED**: bilby now correctly reported as localisation on RHS of aitoff skymaps
- **FIXED**: fix in prob calculations in flattened maps
- **FIXED**: allowing symlinks in plugin folder

v0.4.2 - May 28, 2023

- **ENHANCEMENT**: added a `burst` filtering parameter. If set to True, burst event alerts will pass the filter, otherwise they filtered out (default).
- **REFACTOR**: maps with `CREATOR: ligo-skymap-from-samples` now named on file as `bilby.multiorder.fits`.
- **REFACTOR**: try, except added to listen parser. If an alert breaks the parser an ugly error message is reported, but the listener stays alive to listen to subsequent alerts.
- **FIXED**: burst events causing headaches for filtering and map conversion. The first 'real' burst event provided the necessary alert packet and map to create a realistic unit test for the entire codebase. The code is now much more robust to busrt event alerts.

v0.4.1 - May 25, 2023

- **FIXED**: a bug in filtering routine that caused alerts to pass the filtering algorithm if an unknown parameter in the filtering settings was found. It now fails by default instead of passing.

0.4.0 - May 24, 2023

- **FEATURE**: gocart can now run in daemon mode. Use the commands `gocart listen|quit|restart|status`
- **ENHANCEMENT**: added an `event_dir_exist` parameter to the filtering criteria. If a previous event alert has passed the filtering criteria, it is now possible to allow all subsequent alerts will pass
- **ENHANCEMENT**: redshift range added to the right side of aitoff plots
- **REFACTOR**: removed `ligo.skymap` dependency as this is a huge package and was causing upgrades of gocart to take 20-40 mins. Required new code to convert multi-order skymaps to a single-order (I was previously relying on a function in `ligo.skymap` to do this)

v0.3.0 - May 18, 2023

- **FEATURE**: user can now add their own plugin scripts to run every time an alert is parsed
- **ENHANCEMENT**: added a count of alerts read when first connected to kafka (gives users peace of mind that goacrt is working)
- **FIXED**: area calculations fixed (I was still sort by prob but should have been sorting by probdensity for mulit-order maps)
- **FIXED**: can still read an event ID even if not found in the sky-map header (tried to find the event ID from the map directory path)

v0.2.1 - April 26, 2023

- **FIXED**: listen command was tripping up on mock-events

v0.2.0 - April 26, 2023

- **FEATURE**: filtering of alerts is now possible via options in the settings file (see docs)
- **ENHANCEMENT**: option added to write the original json alert to file
- **ENHANCEMENT**: gocart *should* be robust enough to handle burst events (tested against a hand crafted burst alert packet as none exist in the LVK Public Alert Guide yet).

v0.1.10 - April 21, 2023

- **REFACTOR**: splitting mockevents from superevents

v0.1.9 - April 19, 2023

- **REFACTOR**: sun and moon footprints replace terminator

v0.1.8 - April 6, 2023

- **FIXED**: echo command now parses message to the end of the partition queue
- **FIXED**: listen command remembers where it left off
- **FIXED**: sun & moon coordinates . . . they were not geocentric!

v0.1.7 - April 4, 2023

- **FIXED**: unit test fix

v0.1.6 - April 4, 2023

- **ENHANCEMENT**: added localisation type to aitoff maps
- **FIXED**: first time listen command no longer starteds from time zero but listens from first connection
- **FIXED**: doc builds

v0.1.5 - April 4, 2023

- **FEATURE**: listen command added
- **FEATURE**: echo command added
- **FEATURE**: maps converted to ascii format
- **FEATURE**: maps rendered as aitoff plots
- **FEATURE**: meta.yaml file written with alert, FITS header and extra useful content

v0.1.4 - March 16, 2023

- fixing docsting test

6.6 Modules

`gocart.commonutils`	*common tools used throughout package*
`gocart.convert`	*Convert mulitorder FITS binary tables to other formats*
`gocart.parsers`	*GCN Kafka Notice Parsers*
`gocart.utKit`	*Unit testing tools*

6.6.1 commonutils *(module)*

common tools used throughout package

Functions

flatten_healpix_map(log, mapPath[, nside])	flatten a multiorder healpix map to a specific nside*
generate_skymap_stats(skymap, log)	*Generate some extra stats for a given Healpix map*
getpackagepath()	*Get the root path for this python package*

6.6.2 convert *(module)*

Convert mulitorder FITS binary tables to other formats

Classes

aitoff(log, mapPath, outputFolder[, ...])	*Convert Healpix Map to Aitoff projection with options to include galactic plane and sun position*
ascii(log, mapPath[, nside, settings])	*Take a healpix map and convert to an ascii map with one row per ~deg2*
healpix2cart(log, mapPath[, settings])	*Take a healpix map and convert to a rectilinear projection.*

Functions

create_wcs_and_pixels(log)	*create the all-sky rectilinear wcs*

6.6.3 parsers *(module)*

GCN Kafka Notice Parsers

Classes

lvk(log, record[, settings, plugins])	*The LVK event parser*

6.6.4 utKit *(module)*

Unit testing tools

Classes

`utKit`(moduleDirectory[, dbConn])	*Override dryx utKit*

6.7 Classes

gocart.convert.aitoff	*Convert Healpix Map to Aitoff projection with options to include galactic plane and sun position*
gocart.convert.ascii	*Take a healpix map and convert to an ascii map with one row per ~deg2*
gocart.convert.healpix2cart	*Take a healpix map and convert to a rectilinear projection.*
gocart.parsers.lvk	*The LVK event parser*

6.7.1 aitoff *(class)*

class aitoff(*log*, *mapPath*, *outputFolder*, *settings=False*, *meta={}*, *plotName='skymap.png'*, *patches=None*, *patchesColor='#859900'*, *patchesLabel=None*)

Bases: `object`

Convert Healpix Map to Aitoff projection with options to include galactic plane and sun position

Key Arguments:

- `log` – logger
- `mapPath` – path to the multiorder FITS file
- `outputFolder` – path to output the results to
- `settings` – the settings dictionary
- `meta` – extra meta data to present on plots. Default: {}
- `plotName` – the filename of the plot
- `patches` – a patch collect to add to the plot
- `patchesColor` – colour of the patches. Default '#859900'
- `patchesLabel` – label for patches in the legend. Default None

Usage:

To setup your logger and settings, please use the `fundamentals` package (see tutorial here).

To print the healpix map as an aitoff projection, use the following:

```
from gocart.convert import aitoff
converter = aitoff(
    log=log,
```

(continues on next page)

(continued from previous page)

```
    mapPath="path/to/bayestar.multiorder.fits",
    settings=settings
)
converter.convert()
```

Methods

`convert`([contours, galacticPlane, sunmoon, ...])	*convert the healpix map to an aitoff plot*

convert (*contours=True*, *galacticPlane=True*, *sunmoon=True*, *sunmoonContour=True*)
convert the healpix map to an aitoff plot

Key Arguments:

- `contours` – plot 50 and 90% contours. Default *True*
- `galacticPlane` – plot galactic plane contours. Default *True*
- `sunmoon` – plot sun and moon. Default *True*
- `sunmoonContour` – show contours within 33 deg of sun and 20 deg from moon

Return:

- `plotPath` – path to the printed plot

6.7.2 ascii *(class)*

class ascii (*log*, *mapPath*, *nside=64*, *settings=False*)
Bases: `object`

Take a healpix map and convert to an ascii map with one row per ~deg2

Key Arguments:

- `log` – logger
- `mapPath` – path the the healpix map or an astropy skymap table
- `nside` – size of healpix pixels to resolve the sky to
- `settings` – the settings dictionary

Usage:

To setup your logger and settings, please use the `fundamentals` package (see tutorial here).

To convert a healpix map to ascii, run:

```
from gocart.convert import ascii
c = ascii(
    log=log,
    mapPath="/path/to/bayestar.multiorder.fits",
    nside=64,
    settings=settings
)
asciiContent = c.convert(outputFilepath="/path/to/skymap.csv")
```

Methods

`convert`([outputFilepath])	*Convert the healpix map to ascii format and optionally save the ascii map to file*

convert (*outputFilepath=False*)
Convert the healpix map to ascii format and optionally save the ascii map to file

Key Arguments:

- `outputFilepath` – optionally write content to file. Default *False*

Return:

- `ascii` – the CSV version of the healpix file (nside=64)

6.7.3 healpix2cart *(class)*

class healpix2cart (*log*, *mapPath*, *settings=False*)
Bases: `object`

Take a healpix map and convert to a rectilinear projection.

Key Arguments:

- `log` – logger
- `settings` – the settings dictionary
- `mapPath` – path the the healpix map

Usage:

To setup your logger and settings, please use the `fundamentals` package (see tutorial here).

To initiate a healpix2cart object, use the following:

```
from gocart.convert import healpix2cart
converter = healpix2cart(
    log=log,
    mapPath=pathToOutputDir + "/bayestar.multiorder.fits",
    settings=settings
)
wcs, mapDF, header = converter.convert()
```

Methods

`convert`()	*convert the healpix map to a cartesian WCS and image*

convert ()
convert the healpix map to a cartesian WCS and image

Return:

- `wcs` – an astropy wcs object
- `mapDF` – the map converted cartesian format and recorded in a pandas dataframe.

- `header` – the map header

Usage:

```
from gocart.convert import healpix2cart
converter = healpix2cart(
    log=log,
    mapPath=pathToOutputDir + "/bayestar.multiorder.fits",
    settings=settings
)
wcs, mapDF, header = converter.convert()
```

6.7.4 lvk *(class)*

class lvk(*log*, *record*, *settings=False*, *plugins=False*)

Bases: `object`

The LVK event parser

Key Arguments:

- `log` – logger
- `settings` – the settings dictionary
- `record` – the kafka record to parse.
- `plugins` – run the plugin script found in `~/.config/gocart/plugins` every time an alert is read

Usage:

To setup your logger and settings, please use the `fundamentals` package (see tutorial here).

To parse LVK kafka alerts, use the following:

```
from gocart.parsers import lvk
parser = lvk(
    log=log,
    record=record,
    settings=settings,
    plugins=True
).parse()
```

Methods

`filter_alert`(alert)	*filter the alert record with filtering criteria in the settings file and return true (pass) or false (fail)*
`parse`()	*parse the lvk events and write meta data and maps to file

filter_alert(*alert*)

filter the alert record with filtering criteria in the settings file and return true (pass) or false (fail)

Key Arguments:

- `alert` – the alert record

Return:

- `passing` – True or False. True is alert passes one or more filter

parse()
*parse the lvk events and write meta data and maps to file

6.8 Functions

gocart.commonutils.flatten_healpix_map	flatten a multiorder healpix map to a specific nside*
gocart.commonutils.generate_skymap_stats	*Generate some extra stats for a given Healpix map*
gocart.commonutils.getpackagepath	*Get the root path for this python package*

6.8.1 flatten_healpix_map (*function*)

flatten_healpix_map(*log*, *mapPath*, *nside=64*)
flatten a multiorder healpix map to a specific nside*

Key Arguments:

- `log` – logger
- `mapPath` – path to the multiorder map
- `nside` – the nside index to flatten the map to. Default *64* (~0.9 deg2 pixels)

Return:

- `skymap` – the astropy table of the map

Todo:

- create a sublime snippet for usage
- add a tutorial about `subtract_calibrations` to documentation

```
from gocart.commonutils import flatten_healpix_map
skymap = flatten_healpix_map(
    log=log,
    mapPath=pathToOutputDir + "/bayestar.multiorder.fits",
    nside=64
)
```

6.8.2 generate_skymap_stats (*function*)

generate_skymap_stats(*skymap*, *log*)
Generate some extra stats for a given Healpix map

Key Arguments:

- `skymap` – either a path to a healpix map FITS file or a skymap in astropy table format
- `log` – logger

Return:

- `extras` – a diction of value added map stats

```
from gocart.commonutils import generate_skymap_stats
extras = generate_skymap_stats(
    log=log,
    skymap="path/to/bayestar.multiorder.fits",
)
```

6.8.3 getpackagepath (*function*)

getpackagepath()
Get the root path for this python package

Used in unit testing code

6.9 A-Z Index

PYTHON MODULE INDEX

c

p

u

INDEX