# LI-L2 data reader

This simple command line tool provides an easy and powerful way to get LI data from EUM Data Store and process it by automatically applying the required Light Travel Time (LTT) and parallax correction. The main idea is to apply the necessary corrections all at once on the first read of the data and then save the corrected data locally for any later use. This is especially useful if longer periods of LI data (e.g. days or months) are needed.

## Getting started

1) Just place the files and subdirectories into a directory. About ~2.4GB of disk space is required as the code comes with monthly look-up tables for time and parallax correction.

2) For the EUM Data Store download to work, open the `default.yaml` configuration file and fill in the `consumer_key` and `consumer_secret` in the `eum_dac` section with your personal EUM Data Store credentials.

3) Not absolutely necessary but still highly recommended - use the `./conda_env/mtg_environment.yml` file to create a dedicated conda environment for running the tool. This allows one to create a clean replica of the python environment where the scripts have been successfully tested. Using your own python environment might result in unexpected errors as the `netCDF` module is sensitive to some package inconsistencies that can be avoided by creating the clean conda environment. If new to conda, refer to `Conda_intro.PDF` first.

The two main scripts are `main_li_data_downloader.py` for data download and `main_li_data_reader.py` for processing the downloaded data (including automatic light travel time and parallax correction) and saving the processed data in minutely, hourly or daily netCDF `*.nc` files.

The code has been last tested in `Python 3.12.11` on Linux.

## LI-L2_LFL (Lightning Flash data) example

LI Flash data can be efficiently processed and stored in daily chunks (1-3 million data points per day, up to ~100MB per day). Let's download LI Flashes during the whole day of 2025-03-25:

```
python3 main_li_data_downloader.py l2lfl 20250325
```

The downloaded files should appear in `./downloaded_data/`. Note that the data initially comes in zipped format and also contains many auxiliary files. This script automatically extracts the data and removes all unnecessary files.

Let's now process the downloaded data to take the advantage of the automatic light travel time and parallax correction, and then save (the `-s` argument) the corrected data in a daily netCDF file for any future use:

```
python3 main_li_data_reader.py l2lfl 20250325 -s
```

A new file named `LI-2-LFL-20250325_corrected.nc` should appear in `./processed_data/`, containing all LI Flashes during 2025-03-25 with parallax-corrected latitudes and longitudes, and light-travel time corrected times. On a clean conda environment the command above takes 30-60 seconds to complete (using Dell Latitude 5590, 8GB RAM and Ubuntu 24.04.3).

## LI-L2-LGR (Lightning Group data) example

LI Group data is recommended to be processed and stored in hourly chunks as tens of millions of data points are produced every day and the daily data volume is typically 1-2 GB. This can cause memory problems, if processed all at once. Processing by hour will avoid such issues. Let's download LI Groups during hour 15 UTC on 2025-03-25:

```
python3 main_li_data_downloader.py l2lgr 2025032515
```

The downloaded files should appear in `./downloaded_data/`. Note that the data initially comes in zipped format and also contains many auxiliary files. This script automatically extracts the data and removes all unnecessary files.

Let's now process the downloaded data to take the advantage of the automatic light travel time and parallax correction, and then save (the `-s` argument) the corrected data in a daily netCDF file for any future use:

```
python3 main_li_data_reader.py l2lgr 2025032515 -s
```

A new file named `LI-2-LGR-2025032515_corrected.nc` should appear in `./processed_data/`, containing all LI Groups during hour 15UTC on 2025-03-25 with parallax-corrected latitudes and longitudes, and light-travel time corrected times.

## LI-L2-LEF (Lightning Events in Flashes data) example

LI Lightning Events in Flashes data is recommended to be processed and stored in hourly or even minutely chunks as >100 million data points are produced every day and the daily data volume can reach 10 GB. This can not be processed all at once and processing by hour/minute will avoid memory issues. Let's download LI Lightning Events in flashes during hour 15 UTC on 2025-03-25:

```
python3 main_li_data_downloader.py l2lef 2025032515
```

The downloaded files should appear in `./downloaded_data/`. Note that the data initially comes in zipped format and also contains many auxiliary files. This script automatically extracts the data and removes all unnecessary files.

Let's now process the downloaded data to take the advantage of the automatic light travel time and parallax correction, and then save (the `-s` argument) the corrected data in a daily netCDF file for any future use:

```
python3 main_li_data_reader.py l2lef 2025032515 -s
```

A new file named `LI-2-LEF-2025032515_corrected.nc` should appear in `./processed_data/`, containing all LI Lightning Events in Flashes during hour 15UTC on 2025-03-25 with parallax-corrected latitudes and longitudes, and light-travel time corrected times.

If the hourly processing fails due to memory issues then we can process by minute. To only process the minute 15:45 UTC, run:

```
python3 main_li_data_reader.py l2lef 202503251545 -s
```

A new file named `LI-2-LEF-202503251545_corrected.nc` should appear in `./processed_data/`.

## How to use the time and parallax corrected files?

The data in the light travel time and parallax corrected `*.nc` files created above is stored in a widely known and used `xarray.Dataset` format. There is a small demo script `main_li_data_plotter.py` that allows one to quickly plot or explore the contents of the files.

Let's plot all lightning locations in a file, coloured by the time (hour) of each observation:

```
python3 main_li_data_plotter.py map ./processed_data/LI-2-LFL-20250325_corrected.nc
```

Or we can plot lightning accumulations and optionally add the field-of-view polygons of the LI four cameras (the `-li_fov` argument):

```
python3 main_li_data_plotter.py accmap ./processed_data/LI-2-LFL-20250325_corrected.nc -li_fov
```

The third option is to display some dataset info and demo operations in the terminal:

```
python3 main_li_data_plotter.py print ./processed_data/LI-2-LFL-20250325_corrected.nc
```

For more details and examples you can directly explore the code in `main_li_data_plotter.py`.

## Additional tips and tricks.

1) Parallax correction will always introduce some lightning observations with nan latitude and longitude values. These are the lightning locations along the edges of the visible disk where the LI viewing angle is virtually horizontal and parallax correction becomes infinite. This starts happening ~76 degrees from the sub-satellite point and does not affect EUM members states. However, it can be frequently seen in the west of South America and also in India.

2) The maximum data download and processing period supported by the scripts is 1 day. If a longer period is needed, then a simple command line for-cycle can be very helpful. The commands below download and process LI flashes during 5 days from 20 to 24 March 2025:

```
for dd in {20..24}; do python3 main_li_data_downloader.py l2lfl 202503${dd}; done
```

```
for dd in {20..24}; do python3 main_li_data_reader.py l2lfl 202503${dd} -s; done
```

```
This will result in 5 new daily LI-2-LFL files in `./processed_data/`.
```

3) Consider the `--in_dir` and `--out_dir` options of the data downloader and reader when handling large amounts of data. These allow one to direct the downloaded and corrected data files into a specific directory, e.g. on an external hard drive.

4) If only data over a specific region (e.g. a country) is needed then it is possible to use the `-ll_box` option of the `main_li_data_reader.py`. This will filter the output data for user defined latitude/longitude box, resulting in smaller output files.