PyMef Documentation

Release 1.0.2

Jan Cimbalnik

Aug 02, 2023

Contents

Pymef is a wrapper library for Multiscale Electrophysiology Format developed by MSEL laboratory.

Currently available for all major distributions (Linux, Mac OS, Windows). Only python 3 is supported.

CHAPTER 1

Mef v 3.0 basic features

- Support for parallelisation of signal processing
- Data compression
- Data encryption
- Real-time read/write, failure when writing file leaves intact valid files
- CRC functionality to detect data corruption
- Support for time discontinuities
- Support for time series and video channels

CHAPTER 2

Wrapper features

- MEF3 files write/read
- Convenience functions to easily read data and metadata for multiple channels
- Functions to detect and repair corrupt data

CHAPTER 3

License

Pymef is licensed under the Apache software license.

3.1 Installation

Installation of PyMef is fairly simple. PyMef is built for all major OS (Windows, Mac OS, Linux). However, problems may occur on older systems. Note that PyMef is available for Python 3 only.

To install please use:

```
pip install pymef
```

To install from source:

```
python setup.py install
```

3.2 Getting started

There are two ways one can interact with PyMef wrapper. Either through High level API which is aimed at users with no experience of MEF format. Or through Low level API which is mainly meant for experienced users.

3.2.1 High level API interaction

All operations are handled through *pymef.mef_session.MefSession* class. To initialize a MefSession we can use following code:

```
import numpy
from pymef.mef_session import MefSession

```

(continues on next page)

(continued from previous page)

```
session_path = '/path/to/session.mefd'
password = 'mef_password'

# Either for reading
ms = MefSession(session_path, password)

# Or for writing
ms = MefSession(session_path, password, False, True)
```

Writing time series data

To write time series data the time series metadata file has to be written first.

```
pwd_1 = 'table'
pwd_2 = 'chair'

start_time = 946684800000000
end_time = start_time + 1000000

sampling_frequency = 5000

channel_name = 'test_channel'

# Section 3
section3_dict = {'recording_time_offset': 0,
                 'DST_start_time': 0,
                 'DST_end_time': 0,
                 'GMT_offset': 3600,
                 'subject_name_1': 'Olaf',
                 'subject_name_2': 'Mefson',
                 'subject_ID': '2017',
                 'recording_location': 'pub'}

# Time series section 2 dictionry with ts section 2 fields
section2_ts_dict = {'channel_description': 'Test_channel',
                    'session_description': 'Test_session',
                    'recording_duration': 1,
                    'reference_description': 'wine',
                    'acquisition_channel_number': 5,
                    'sampling_frequency': sampling_frequency,
                    'notch_filter_frequency_setting': 50.0,
                    'low_frequency_filter_setting': 1.0,
                    'high_frequency_filter_setting': 10.0,
                    'AC_line_frequency': 70,
                    'units_conversion_factor': 1.5,
                    'units_description': 'uV',
                    'start_sample': 0,  # Different for segments
                    'number_of_discontinuities': 1,
                    # The following entries are filled automatically during data␣
↪writing
                    'maximum_native_sample_value': 0.0,
                    'minimum_native_sample_value': 0.0,
                    'number_of_blocks': 0,
                    'maximum_block_bytes': 0,
                    'maximum_block_samples': 0,
```

(continues on next page)

(continued from previous page)

```
                      'maximum_difference_bytes': 0,
                      'block_interval': 0,
                      'maximum_contiguous_blocks': 0,
                      'maximum_contiguous_block_bytes': 0,
                      'maximum_contiguous_samples': 0,
                      'number_of_samples': 0}

ms.write_mef_ts_segment_metadata(channel_name,
                                 segment_n,
                                 pwd_1,
                                 pwd_2,
                                 start_time,
                                 end_time,
                                 section2_ts_dict,
                                 section3_dict)
```

After the metadata file is in place the data itslef can be written:

```
N = 20000  # Number of samples
samps_per_mef_block = 5000
data = np.random.randint(-200, 200, N, dtype='int32')

ms.write_mef_ts_segment_data(channel_name,
                             0,
                             pwd_1,
                             pwd_2,
                             samps_per_mef_block,
                             raw_data)
```

Reading time series data

To obtain information about time series channels the function *read_ts_channel_basic_info()* as in the following code snippet.

```
ms.read_ts_channel_basic_info()
```

There are two options for reading time series data:

- By sample with function *read_ts_channels_sample()*
- By uUTC with function *read_ts_channels_uutc()*

Both functions have similar API. The first argument is either a channel string or a list of channel strings. The secund argument consits of list of lists with start and stop sample or uutc. Both time entries can also be None in which case the recording start / stop is used. If only one entry for start and stop is provided the same span is applied to all channels in the channel list. There is one minor difference in returned data - when reading by uUTC the function returns NaNs if no data is available in the specified time span.

```
# Returns 1D numpy array with data from recording start to recording stop
ms.read_ts_channels_uutc(channel, [[None, None]])

# Returns 1D numpy array with 2 1D numpy arrays with data from recording start to␣
↪recording stop
ms.read_ts_channels_uutc([channel, channel], [[None, None]])
```

(continues on next page)

(continued from previous page)

```
# Returns 1D numpy array with data from sample 5 to sample 5000
ms.read_ts_channels_sample(channel, [[5, 5000]])
```

3.3 PyMef API

PyMef api consist of two parts. - High level API which should be used in most cases - Low level API which is aimed at users experienced with MEF library

3.3.1 High level API

class pymef.mef_session.**MefSession**(*session_path*, *password*, *read_metadata=True*, *new_session=False*, *check_all_passwords=True*)

Basic object for operations with mef sessions.

Parameters

- **session_path** (*str*) – path to mef session
- **password** (*str*) – password for mef session
- **read_metadata** (*bool*) – whether to read metadata (default=True)
- **new_session** (*bool*) – whether this is a new session for writing (default=False)
- **check_all_passwords** (*bool*) – check all files or just the first one encoutered(default=True)

annonymize_session(*password_1*, *password_2*, *new_name=None*, *new_id=None*)

Anonymize mef session

Parameters

- **password_1** (*str*) – Level 1 password
- **password_2** (*str*) – Level 2 password
- **new_name** (*str*) – New first name for the subject (default = None)
- **new_id** (*str*) – New subject id (default = None)

Returns **result** – None on success

Return type object

append_mef_ts_segment_data(*channel*, *segment_n*, *password_1*, *password_2*, *start_time*, *end_time*, *samps_per_mef_block*, *data*, *discontinuity_flag=False*)

Appends new time series metadata in the specified segment

Parameters

- **channel** (*str*) – Channel name
- **segment_n** (*int*) – Segment number
- **password_1** (*str*) – Level 1 password
- **password_2** (*str*) – Level 2 password
- **start_time** (*int*) – Start time of the appended data

- **end_time** – End time of the appended data
- **samps_per_mef_block** (*int*) – Number of samples per mef block
- **data** (*np.array*) – 1-D numpy array of type int32
- **discontinuity_flag** (*bool*) – Discontinuity flag for appended data.

change_channel_name (*change_dict*)
Chnges name of one or more channels

Parameters **change_dict** (*dict*) – Dictionary with channels to be changed {old_name: new_name}

Returns **result** – None on success

Return type object

create_slice_session (*slice_session_path*, *slice_start_stop*, *password_1*, *password_2*, *samps_per_mef_block=None*, *time_unit='uutc'*)
Function to create slice of the mef session (time series only).

Parameters

- **slice_session_path** (*str*) – Path to new sliced session (including .mefd suffix)
- **slice_start_stop** (*list*) – List containing 2 int with start and stop uutc times of the slice
- **password_1** (*str*) – Level 1 password
- **password_2** (*str*) – Level 2 password
- **samps_per_mef_block** (*int*) – Number of samples for mef block. Default=channel sampling frequency

detect_corrupt_data (*repair=False*)
Detects corrupt data

Parameters **repair** (*bool*) – Whether to try to repair data (default=False)

Returns **result** – None on success

Return type object

get_channel_toc (*channel*)
Returns discontinuities accross segments.

Parameters **channel** (*str*) – Channel to calculate TOC on

Returns

TOC –

Array with

- [0,:] = discontinuity flags
- [1,:] = n block samples
- [2,:] = start samples
- [3,:] = start uutc times

Return type np.array

read_records (*channel=None*, *segment_n=None*)
Returns list of dictionaries with MEF records.

Parameters

- **channel** (*str*) – Session channel, if not specified, session records will be read (default = None)
- **segment_n** (*int*) – Segment number, if not specified, channel records will be read (default = None)

Returns **record_list** – List of dictionaries with record entries

Return type list

read_ts_channel_basic_info()
Reads session time series channel names

Returns

channel_list –

List of dictionaries with information about channels:

- Sampling frequency
- Number of samples
- Units conversion factor
- Units description
- Earliest start time
- Latest end time
- Channel description

Return type list

read_ts_channels_sample(*channel_map*, *sample_map*, *process_n=None*)
Reads desired channels in desired sample segment

Parameters

- **channel_map** (*str or list*) – Channel or list of channels to be read
- **sample_map** (*list*) – List of [start, stop] samples to be loaded that correspond to channel_map. if there is only one entry the same range is applied to all channels
- **process_n** (*int*) – How many processes use for reading (default=None)

Returns **data** – Numpy array of numpy array objects [channels,samples] or 1D numpy array

Return type np.array(dtype=np.float32)

read_ts_channels_uutc(*channel_map*, *uutc_map*, *process_n=None*, *out_nans=True*)
Reads desired channels in desired time segment. Missing data at discontinuities are filled with NaNs.

Parameters

- **channel_map** (*str or list*) – Channel or list of channels to be read
- **uutc_map** (*list*) – List of [start,stop] uutc times to be loaded that correspond to channel_map. if there is only one entry the same range is applied to all channels
- **process_n** (*int*) – How many processes use for reading (defualt = None)
- **out_nans** (*bool*) – Whether to return an array of np.nan if the uutc times for channel are completely out of start and end times

Returns **data** – Numpy array of numpy array objects [channels,samples] or 1D numpy array

Return type np.array(dtype=np.float32)

write_mef_records(*password_1*, *password_2*, *start_time*, *end_time*, *time_offset*, *records_list*, *channel_type='ts'*, *channel=None*, *segment_n=None*)

Writes new records on session level. If channel is specified, the the records are written at channel level. If segment_n is sepcified, the recordes are written at segment level.

Parameters

- **password_1** (*str*) – Level 1 password
- **password_2** (*str*) – Level 2 password
- **start_time** (*int*) – Records start time
- **time** (*end*) – Records end time
- **time_offset** (*int*) – Time offset for records
- **record_list** (*list*) – List with record dictionaries
- **channel_type** (*str*) – Channel type: 'ts' (time series) or 'v' (video), default='ts'
- **channel** (*str*) – Channel name
- **segment_n** (*int*) – Segment number

Notes

Each entry in record list must be a dictionary and contain the field "type". The rest of the entries are optional. All times are in uUTC or us. The following types are recognized:

Note - simple note:

- type - "Note"
- time - record time (int)
- text - note text (str)

SyLg - system log:

- type - "SyLg"
- time - record time (int)
- text - system log text

EDFA - EDF annotation record:

- type - "EDFA"
- time - record time (int)
- duration - duration of the event (int)
- text - annotation text (str)

LNTP - Line noise template:

- type - "LNTP"
- time - record time (int)
- length - template length (int)
- template - 1D numpy array of template itself (np.arry, dtype=int)

CSti - Cognitive stimulation:

- type - "CSti"
- time - record time (int)
- task_type - type of the task (str)
- stimulus_duration - duration of the stimulus (int)
- stimulus_type - type of the stimulus (str)
- patient_response - response of the patient (str)

ESti - Electrical stimulation:

- type - "Esti"
- time - record time (int)
- amplitude - stimulus amplitude (float)
- frequency - frequency of the stimulus (float)
- pulse_width - pulse width (int)
- **ampunit_code - code of amplitude unit (int)**
  - -1 = no entry
  - 0 = unknown
  - 1 = mA
  - 2 = V
- **mode code - code of the stimulation mode (int)**
  - -1 = no entry
  - 0 = unknown
  - 1 = current
  - 2 = voltage
- anode - stimulation anode (str)
- catode - stimulation catode (str)

Seiz - Seizure:

- type - "Seiz"
- time - record time (int)
- earliest_onset - earliest uUTC onset of the seizure (int)
- latest offset - latest uUTC offset of the seizure (int)
- duration - duration of the seizure (int)
- number_of_channels - number of seizure onset channels (int)
- **onset_code - code for the onset**
  - -1 = no entry
  - 0 - unknown
  - 1 - docal

  - 2 - generalized
  - 3 -propagated
  - 4 - mixed
- marker_name_1 - name of the marker 1 (str)
- marker_name_2 - name of the marker 2 (str)
- annotation - seizure annotation (str)
- **channels - list of dictionaries with channel entries**
  - name - name of the channel (str)
  - onset - seizure onset on this channel (int)
  - offset - seizure offset on this channel (int)

Curs - Cursor record:

- type - "Curs"
- time - record time (int)
- id_record - id of the record (int)
- trace_timestamp - timestamp of the trace (int8)
- latency - latency of the record (int)
- name - name of the cursor record (str)

Epoc - Epoch record:

- type - "Epoc"
- time - record time (int)
- id_record - id of the record (int)
- timestamp - start timestamp of the epoch (int8)
- end_timestamp - end timestamp of the epoch (int8)
- duration - duration of the epoch
- type - epoch type (str)
- text - descriptive text of the epoch (str)

write_mef_ts_segment_data(*channel*, *segment_n*, *password_1*, *password_2*, *samps_per_mef_block*, *data*)

Writes new time series data in the specified segment

Parameters

- **channel** (*str*) – Channel name
- **segment_n** (*int*) – Segment number
- **password_1** (*str*) – Level 1 password
- **password_2** (*str*) – Level 2 password
- **samps_per_mef_block** (*int*) – Number of samples per mef block
- **data** (*np.array*) – 1-D numpy array of type int32

write_mef_ts_segment_metadata(*channel*, *segment_n*, *password_1*, *password_2*, *start_time*, *end_time*, *section_2_dict*, *section_3_dict*)

Writes new time series metadata in the specified segment

Parameters

- **channel** (*str*) – Channel name
- **segment_n** (*int*) – Segment number
- **password_1** (*str*) – Level 1 password
- **password_2** (*str*) – Level 2 password
- **start_time** (*int*) – Start time
- **time** (*end*) – End time
- **section_2_dict** (*dict*) – Dictionary with user specified section_2 fileds
- **section_3_dict** (*dict*) – Dictionary with user specified section_3 fileds

write_mef_v_segment_indices(*channel*, *segment_n*, *password_1*, *password_2*, *start_time*, *end_time*, *index_entries*)

Writes new video indices in the specified segment

Parameters

- **channel** (*str*) – Channel name
- **segment_n** (*int*) – Segment number
- **password_1** (*str*) – Level 1 password
- **password_2** (*str*) – Level 2 password
- **start_time** (*int*) – Start time of video indices
- **time** (*end*) – End time of video indices
- **index_entries** (*np.array*) – Numpy array with vi_dtype

write_mef_v_segment_metadata(*channel*, *segment_n*, *password_1*, *password_2*, *start_time*, *end_time*, *section_2_dict*, *section_3_dict*)

Writes new video metadata in the specified segment

Parameters

- **channel** (*str*) – Channel name
- **segment_n** (*int*) – Segment number
- **password_1** (*str*) – Level 1 password
- **password_2** (*str*) – Level 2 password
- **start_time** (*int*) – Start time of the data
- **time** (*end*) – End time of the data
- **section_2_dict** (*dict*) – Dictionary with user specified section_2 fileds
- **section_3_dict** (*dict*) – Dictionary with user specified section_3 fileds

3.3.2 Low level API

Warning: Use with caution! Improper use may lead to data corruption.

This submodule provides a wrapper around Mayo Electrophysiology Format (MEF) version 3.0 library.

pymef.mef_file.pymef3_file.**append_ts_data_and_indices**()
Function to append MEF3 time series data and indices to existing segment files.

Parameters

- **target_path** (*str*) – Path to segment being appended
- **password_1** (*str*) – Level 1 password.
- **password_2** (*str*) – Level 2 password.
- **samples_per_mef_block** (*int*) – Number of samples in one MEF RED block.
- **raw_data** (*np.array*) – Numpy 1D array with raw data of dtype int32.
- **discontinuity_flag** (*bool*) – Flag to mark discontinuity at the start of appended data (default=True)
- **lossy_flag** (*bool*) – Flag for optional lossy compression (default=False).

pymef.mef_file.pymef3_file.**check_mef_password**()
Function to check MEF3 password validity.

Parameters

- **target_path** (*str*) – Path to MEF3 metadata file.
- **password** (*str*) – Level 1 or level 2 password.

Returns

password_type –

- 0 - incorrect password
- 1 - level 1 password
- 2 - level 2 password

Return type int

pymef.mef_file.pymef3_file.**read_mef_channel_metadata**()
Function to read MEF3 channel metadata.

Parameters

- **target_path** (*str*) – Path to MEF3 channel being read.
- **password** (*str*) – Level 1 or level 2 password.
- **map_indices_flag** (*bool*) – Flag to enable the mapping of the time-series and video indices (default=True, map indices)
- **copy_metadata_to_dict** (*bool*) – Flag to copy metadata into a python dictionary structure (True), instead of returning the metadata by reference in Numpy structured datatypes (Default=False)

Returns **channel_metadata** – Dictionary with channel metadata and all segments metadata and records.

Return type dict

pymef.mef_file.pymef3_file.**read_mef_segment_metadata**()
Function to read MEF3 segment metadata.

Parameters

- **target_path** (*str*) – Path to MEF3 segment being read.
- **password** (*str*) – Level 1 or level 2 password.
- **map_indices_flag** (*bool*) – Flag to enable the mapping of the time-series and video indices (default=True, map indices)
- **copy_metadata_to_dict** (*bool*) – Flag to copy metadata into a python dictionary structure (True), instead of returning the metadata by reference in Numpy structured datatypes (Default=False)

Returns **segment_metadata** – Dictionary with segment metadata and records.

Return type dict

pymef.mef_file.pymef3_file.**read_mef_session_metadata**()
Function to read MEF3 session metadata.

Parameters

- **target_path** (*str*) – Path to MEF3 session being read
- **password** (*str*) – Level 1 or level 2 password.
- **map_indices_flag** (*bool*) – Flag to enable the mapping of the time-series and video indices (default=True, map indices)
- **copy_metadata_to_dict** (*bool*) – Flag to copy metadata into a python dictionary structure (True), instead of returning the metadata by reference in Numpy structured datatypes (Default=False)

Returns **session_metadata** – Dictionary with session metadata and all channels and segments metadata and records.

Return type dict

pymef.mef_file.pymef3_file.**read_mef_ts_data**()
Function to read MEF3 time series data.

Parameters

- **channel_specific_metadata** (*np.ndarray*) – Channel metadata
- **start** (*int*) – Start sample or uUTC time to be read.
- **end** (*int*) – End sample or uUTC time to be read.
- **time_flag** (*bool*) – Flag to indicate if user is reading by samples or uUTC times (default=False - reading by sample)

Returns **data** – 1D numpy array (dtype=float) with data. If the data is read by uUTC and a gap is present the missing values are filled with NaNs

Return type np.array

pymef.mef_file.pymef3_file.**write_mef_data_records**()
Function for writing MEF3 records at any level specified by path parameter.

Parameters

- **target_path** (*str*) – Path where record files will be written.
- **password_1** (*str*) – Level 1 password.
- **password_2** (*str*) – Level 2 password.
- **start_time** (*int*) – uUTC start time for universal header.
- **end_time** (*int*) – uUTC end time for universal header.
- **recording_offset** (*int*) – Offset for uUTC times.
- **record_list** (*list*) – List of record dictionaries consisting of numpy arrays.

pymef.mef_file.pymef3_file.**write_mef_ts_data_and_indices**()
Function to write MEF3 time series data and indices file.

Parameters

- **target_path** (*str*) – Path to segment being written.
- **password_1** (*str*) – Level 1 password.
- **password_2** (*str*) – Level 2 password.
- **samples_per_mef_block** (*int*) – Number of samples in one MEF RED block.
- **raw_data** (*np.array*) – Numpy 1D array with raw data of dtype int32.
- **lossy_flag** (*bool*) – Flag for optional lossy compression (default=False).

pymef.mef_file.pymef3_file.**write_mef_ts_metadata**()
Function to write MEF3 time series metadata file.

Parameters

- **target_path** (*str*) – Path to segment being written.
- **password_1** (*str*) – Level 1 password.
- **password_2** (*str*) – Level 2 password.
- **start_time** (*int*) – uUTC recording start time - also used as recording time offset.
- **end_time** (*int*) – uUTC recording end time.
- **section_2_arr** (*np.array*) – Numpy array of time series metadata section 2 dtype.
- **section_3_arr** (*np.array*) – Numpy array of metadata section 3 dtype.

pymef.mef_file.pymef3_file.**write_mef_v_indices**()
Function to write MEF3 video indices file.

Parameters

- **target_path** (*str*) – Path to segment being written.
- **password_1** (*str*) – Level 1 password.
- **password_2** (*str*) – Level 2 password.
- **start_time** (*int*) – uUTC recording start time - minimum value of index entries.
- **end_time** (*int*) – uUTC recording end time - maximum value of index entries.
- **index_entries** (*list*) – List of numpy arrays with index entries.

pymef.mef_file.pymef3_file.**write_mef_v_metadata**()
Function to write MEF3 video metadata file.

Parameters

- **target_path** (*str*) – Path to segment being written.
- **password_1** (*str*) – Level 1 password.
- **password_2** (*str*) – Level 2 password.
- **start_time** (*int*) – uUTC recording start time - also used as recording time offset.
- **end_time** (*int*) – uUTC recording end time.
- **section_2_arr** (*np.array*) – Numpy array of video metadata section 2 dtype.
- **section_3_arr** (*np.array*) – Numpy array of metadata section 3 dtype.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index