Epycon: A Single-platform Python Package for Parsing and Converting Raw Electrophysiology Data into Open Formats

Jakub Hejc[1,2,3], Richard Redina[1,3], Jana Kolarova[3], Zdenek Starek[1,4]

[1] International Clinical Research Center, St. Anne's University Hospital, Brno, Czech Republic
[2] Department of Pediatric, Children's Hospital, The University Hospital Brno, Brno, Czech Republic
[3] Department of Biomedical Engineering, Brno University of Technology, Brno, Czech Republic
[4] 1st Department of Internal Medicine, Cardio-Angiology, Faculty of Medicine, Masaryk University, Brno, Czech Republic

Abstract

The WorkMate™ platform is a widely used system for recording and analyzing cardiac electrophysiology waveforms recorded during interventional procedures. However, the internal export tools do not usually allow full access to unprocessed data, which challenges its use for research purposes.

The Epycon package is a data parsing tool that enables direct access to raw unipolar/bipolar waveforms and clinical annotations stored in WorkMate™ proprietary serialized file formats. It provides the ability to convert the data into established open formats, including Comma-Separated Values and Hierarchical Data Format 5. Epycon offers a number of features that make it a valuable tool for researchers, including memory-efficient batch processing, support for both raw unipolar and bipolar waveforms, and comprehensive documentation of WorkMate™ file format for development purposes.

Epycon is implemented in Python 3.8 and can help researchers to more efficiently access electrophysiology data, facilitating research in developing algorithms for cardiac electrogram processing or creating open-source databases of electrophysiology data for public use.

1. Introduction

Interventional electrophysiology (EP) procedures play a crucial role in the effective diagnosis and treatment of cardiac arrhythmias [1]. It is essential for investigating the electrical properties of the heart and gaining insights into the arrhythmia mechanism [2].

Recent data from the United States reports a growing trend in the number of EP procedures with the annual rate surging from approximately 320 to 675 procedures per 100,000 medical beneficiaries [3]. The trend has resulted in a substantial reservoir of clinical recordings that prove invaluable to various research fields.

EP studies typically encompass the recording of intracardiac electrograms, the implementation of pacing protocols to observe conduction patterns under stress conditions, and the measurement of local activation intervals [1, 4]. They are supported by an infrastructure comprising complex hardware and software systems designed specifically for gathering, display, processing, analysis and storage of the procedural data [4].

The acquired waveforms are usually stored in closed, platform-dependent file formats with dedicated data transfer and exportation tools supporting general needs of clinical practitioners. A file format documentation is seldom provided by the manufacturers, which poses a significant obstacle to efficiently extracting and collecting the study data. Researchers seeking access to complete raw recordings must rely on dedicated tools constrained by internal settings and time-consuming manual adjustment.

The WorkMate™ and WorkMate Claris™ Systems (Abbott Laboratories, Abbott Park, IL, USA) belong among the widespread acquisition and processing units utilized to support EP procedures. Our aims are to provide comprehensive file format documentation for development purposes, along with a custom parser for Python that complements exportation tools provided by the platform.

2. Methods

2.1. System and data description

The WorkMate™ EP platform facilitates recording of the waveforms through a proprietary amplifier while employing minimal built-in hardware filtering. The waveforms can be sampled at frequencies of up to 2000 Hz, with a resolution of 78 nV/LSb. Post-acquisition filtering is performed using digital filters. Additionally, bipolar intracardiac channels are derived from the original unipolar

recordings. The system has the capacity to record up to 256 channels. Up to overall 448 channels can be configured and adjusted by the system operator.

The waveforms are stored as raw signals into a separate data file (DLog) per each recording session matching an expression `^[0-9]{8}\.log$`. The file `entries.log` (EntriesLog) contains clinical and procedural annotations provided by the operator or by the system's annotation algorithms.

The waveforms are stored as raw signals in separate data files (DLog) for each recording session, matching an expression `^[0-9]{8}.log$`. The file `entries.log` (EntryLog) contains clinical and procedural annotations provided by the operator or by the system's internal annotation algorithms. Other study-specific files have been excluded from our analysis.

2.2. File format decoding

The contiguous arrays of bytes in the DLog and EntryLog was decoded through visual reverse engineering and inter-file differential analysis. This process involved identifying byte blocks that contained raw data chunks, acquisition settings, processing and visualization settings, and clinical annotations.

In general, we established a set of demographic, procedural, acquisition, and processing parameters, denoted as $P = \{p_1, p_2, \ldots, p_i\}$, which were known to have a functional connection to the data and were expected to be integral components of the analyzed files. For each $p_i \in P$, the set of potential parameter values $S_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,j}\}$ was determined by considering the acceptable range and step size allowed by the platform for the given parameter. We illustrate the process of identifying the byte addresses for each p_i using pseudocode in Algorithm 1.

Algorithm 1 Differential analysis

Input: Sets P and S_i
Output: Start and stop bytes $\{b_{i,1}, b_{i,2}\}$ for each p_i

for each $p_i \in P$ **do**
  for each $s_{i,j} \in S_i$ **do**
    Set p_i to $s_{i,j}$
    $\mathrm{D_{i,j}} \longleftarrow$ Acquire and store data
    Find each $[b_{i,1}$ to $\mathrm{b_{i,2}}]$ where $\mathrm{D_{i,j-1}} \neq \mathrm{D_{i,j}}$
    $\mathrm{d_{i,j}} \longleftarrow$ Decode $\mathrm{D_{i,j}}$ $[\mathrm{b_{i,1}}$ to $\mathrm{b_{i,2}}]$
    if $\mathrm{d_{i,j}} = \mathrm{s_{i,j}}$ **then** Collect $\{b_{i,1}, b_{i,2}, p_i\}$
    end if
  end for
  Select best $\{b_{i,1}, b_{i,2}, p_i\}$
end for

2.3. Data model and formats

The package utilizes the h5py library to export the data into Hierarchical Data Format 5 (HDF5) file [5]. In order to simplify and streamline data access operations, we implemented a custom API wrapper on top of h5py. The wrapper provides more intuitive interface for working with our data model ensuring compatibility with a third-party signal processing and viewing software SignalPlant [6]. The data model incorporates four mandatory datasets within the HDF5 file structure specified in Table 1.

Table 1. HDF5 file structure. Attr – attribute; DS – data set; FP – floating point; SArray – structured array composed of multiple primitive data types; C – number of channels; N – number of samples.

Item	Item name	Type	Description
Attr	Fs	FP32	Sampling frequency
Attr	GeneratedBy	String	Export metadata
Attr	LeftI	FP64	–
Attr	RightI	FP64	–
DS	ChannelSettings	SArray	Display settings
DS	Data	FP32	CxN Waveform data
DS	Info	SArray	Channel name/unit
DS	Marks	SArray	Annotation + location

Alternatively, the package uses a simple columnar layout with one column per data field to store the waveforms in the Comma-separated text (CSV) format [7].

Tabular data are accompanied by a separate SEL text file. The SEL file provides encapsulation of the metadata and annotation marks and ensures these data can be visualized in the SignalPlant platform.

2.4. Implementation and dependencies

The package was implemented in Python[8] 3.8. Versions supported for Epycon 1.0.1 release are $3.8 - 3.10$. Supported versions of the EP platforms are WorkMate™ $\geq$ 4.1 and WorkMate Claris™ 1.1.1. External dependencies include Numpy 1.23.1 and h5py 3.6.0.

3. Results

3.1. Main features

This section presents components of the Epycon package and the outcomes of our investigation into format decoding. Figure 1 shows a high-level representation of the package architecture.

The parser core offers classes designed for the independent parsing of waveforms, annotation marks, and metadata. The application interface (API) exposes core classes

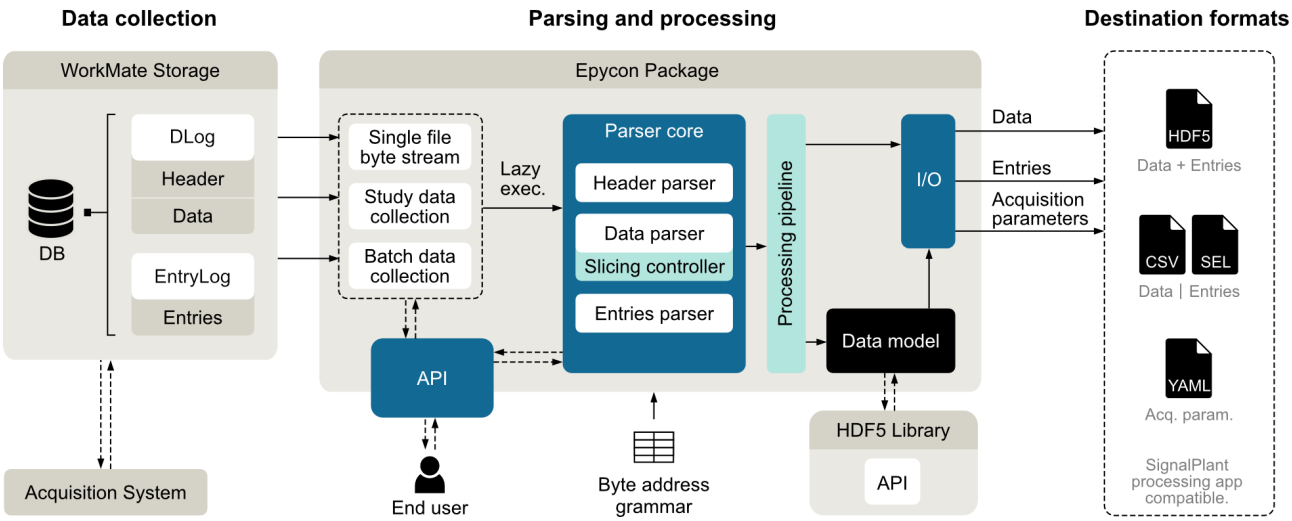

Figure 1. A high-level representation of the package components and dependencies. Individual parsers are exposed via application interface (API) to provide customized data handling.

to users, allowing them to customize the data handling process. Users can access designated sections of raw data, employing option for sample-wise and channel-wise slicing.

By default, the waveform parser reads the entire data block into memory prior to a conversion. Users have the flexibility to define a custom chunk size for more memory-efficient conversion or to employ a lazy-loading parser for customized solutions.

The package include versatile platform for batch processing, enabling users to operate on the entire database, specific EP studies, or selected files, thereby accommodating various analytical needs.

3.2. File format decoding

Figures 2 and 3 document the structural and compositional aspects of the DFile and EntryFile binary formats, respectively. The initial row in each schema represents the beginning of the byte stream. Individual data types are comprehensively documented within the source code.

Each recording session creates a separate DFile containing a header block with acquisition parameters such as sampling frequency, resolution, types of hardware filters, channel settings, and electrode pairings, and a data block containing chunks of waveform data. The EntryFile provides information about each annotation mark created during the EP procedure, including the annotation type, time of creation, and name of the related DFile.

3.3. Performance evaluation

A single-thread read-write (R-W) performance has been tested on Intel® Xeon® X5650 2.67GHz 12MB Cache, 48MB 1333 MHz DDR3 ECC, PCie 2.0 NVMe SSD Kingston NV1. We compared the R-W performance in terms of conversion time and relative file size change without using compression for data from 12 animal EP procedures performed with identical system settings including number of acquired channels.

Header	14 652 B	UNIX time stamp	2×4 B
		No. of active channels	2 B
		Unknown bytes	2 B
		Channels settings	14 336 B
		Acquisition parameters	4×2 B
		Mapping of channel ID to sample address	2 B
		Data chunk offset	2 B
Data	N×M×4 B	Data chunk 0	M×4 B
		…	
		Data chunk N-1	M×4 B

0	
32 B	
Name	12 B
Scale	2×1 B
UID	2×1 B
Filters	2×2 B
Color	1 B
Unknown	1 B
Pins	2×1 B
Analysis	2×1 B
…	
447	
32 B	

Figure 2. Schema documentation of the DFile internal serialized binary format. M: number of active channels; N: number of data samples.

Header	36 B	Unknown	2 B
		UNIX time stamp	2×4 B
		Study date: ASCII	10 B
Annotations		Annotation 0	220 B
		Annotation 1	220 B
		…	
		Annotation C-1	220 B

0	
220 B	
Unknown	2 B
File ID	4 B
Unknown	4 B
UNIX time stamp	2×4 B
Color	1 B

Figure 3. Schema documentation of the EntryFile internal serialized binary format. C: number of annotations.

The mean size of the study was 4.42±2.63 GB. The mean conversion time per study was 13.9±8.0 and 178.4±106.6 s for HDF5 and CSV, respectively. The post-conversion relative change in file size with preserved 32-bit precision was +2.1% for HDF5 and +97.0% in case of CSV. Overall conversion rates were 574 MBps (HDF5) and 51 MBps (CSV).

4. Conclusion

The Epycon package offers memory-efficient batch processing, direct access to raw unipolar/bipolar waveforms stored in proprietary binary format, and the ability to convert the waveforms and metadata into established open formats, including CSV and HDF5.

5. How to use the package

The package is available via pip or on GitHub repository github.com/fnusa-icrc-ice/epycon.

Following command can be used to convert both waveforms and annotation marks into HDF5 with optional chunk size using the command line interface:

```
$ python -m epycon -i <study folder> \
    -o <output folder> --marks \
    --format hdf5 --chunk-size 64
```

The package can be run with YAML configuration file to specify more comprehensive options. For example:

```
YAML
input:
    folder: <path to folder>
    studies: [<study id>, ]
output:
    folder: <path to folder>
    file_format: <str>
processing:
    select_channels: [<channel name>, ]
    mount: <bool>
...
```

Once you have edited the YAML file, you can run the package with the config file using the following command:

```
$ python -m epycon -c config.yaml
```

The following example shows how to use Epycon API in own code to lazily read waveform segment:

```
from epycon.core import LogDataParser

with LogDataParser(file_path,
    chunksize, start_sample, end_sample,
) as parser:
    next(parser)
```

The presented examples are for illustrative purposes only. The syntax and behavior of the package may change in future releases.

Acknowledgments

This publication was written at Masaryk University as part of the project "Novel imaging, computing and analytical methods in cardiovascular diseases diagnostics and monitoring" number MUNI/A/1410/2022 with the support of the Specific University Research Grant, as provided by the Ministry of Education, Youth and Sports of the Czech Republic in the year 2023.

Brno Ph.D. Talent Scholarship Holder R.R. funded by the Brno City Municipality.

References

[1] Katritsis DG, Boriani G, Cosio FG, Hindricks G, Jaïs P, Josephson ME, et al. European heart rhythm association (EHRA) consensus document on the management of supraventricular arrhythmias, endorsed by heart rhythm society (HRS), asia-pacific heart rhythm society (APHRS), and sociedad latinoamericana de estimulación cardiaca y electrofisiologia (SOLAECE). EP Europace November 2016; 19(3):465–511.

[2] Issa ZF, Miller JM, Zipes DP. Electrophysiological testing. In Clinical Arrhythmology and Electrophysiology. Elsevier, 2019; 81–124.

[3] Scott M, Baykaner T, Bunch TJ, Piccini JP, Russo AM, Tzou WS, Zeitler EP, Steinberg BA. Contemporary trends in cardiac electrophysiology procedures in the united states, and impact of a global pandemic. Heart Rhythm O2 March 2023; 4(3):193–199.

[4] Haines DE, Beheiry S, Akar JG, Baker JL, Beinborn D, Beshai JF, et al. Heart rhythm society expert consensus statement on electrophysiology laboratory standards: Process, protocols, equipment, personnel, and safety. Heart Rhythm August 2014;11(8):e9–e51.

[5] The HDF Group. Hierarchical Data Format, version 5, 1997-2023. URL `http://www.hdfgroup.org/HDF5`.

[6] Plesinger F, Jurco J, Halamek J, Jurak P. SignalPlant: an open signal processing software platform. Physiological Measurement May 2016;37(7):N38–N48.

[7] Shafranovich Y. Common format and MIME type for comma-separated values (CSV) files. Technical report, October 2005.

[8] Van Rossum G, Drake FL. Python 3 Reference Manual. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

Address for correspondence:

Jakub Hejc
ICRC – St. Anne's University Hospital, Pekarska 53, 602 00 Brno, Czech Republic
jakub.hejc@fnusa.cz