OpenXDF Documentation

Ryan Opel - opelr@ohsu.edu

2019-01-02 – v0.5.1

Overview

`openxdf` is a Python module built for interacting with Open eXchange Data Format files.

`.xdf` files are XML-formatted header files that provide all of the information necessary to interpret a signal data file for a polysomnogram (PSG; sleep study). This module gives users simple methods of accessing the data stored in these documents, and helps associate the header information stored in `.xdf` files together with the raw data it"s referencing.

Installation

Users can install `openxdf` with `pip` by running the command `pip install openxdf`.

Note: Python 3.6 or greater is required to install openxdf.

Use

`openxdf` is centered around two class objects: openxdf.OpenXDF, which acts as a wrapper for `.xdf` files, and openxdf.Signal, which wraps the raw signal-data files.

An example:

```
>>> import openxdf
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.header
{"ID": "Example", "EpochLength": 30, "FrameLength": 1, "Endian": "little",
"File": "example.data"}
>>> xdf.sources
[{"SourceName": "FP1", "Unit": 1e-06, "UseGridScale": "false",
  "MinSamplingRate": 200, "MinSampleWidth": 1, "Ignore": "false",
  "PhysicalMax": 3199.9, "Signed": "true", "SampleWidth": 2,
  "SampleFrequency": 200, "DigitalMax": 32767, "DigitalMin": -32768,
  "PhysicalMin": -3200, "DigitalToVolts": 0.0976563},
  {...},
]

>>> signal = openxdf.Signal(xdf, "/path/to/file/.../example.data")
>>> signal.list_channels
["EOG-L", "EOG-R", "F3-A2", "F4-A1", "C3-A2", "C4-A1", "O1-A2",
 "O2-A1", ...]
>>> signal.read_file(["EOG-L", "C4-A1"])
{'EOG-L': array([[-890, -885, -803, ...,  393,  440,  422],
                 [ 494,  396,  451, ...,  323,  338,  420],
                 [ 504,  439,  493, ...,  251,  300,  244],
                 ...,
                 [  47, -104,  -79, ...,    9, -149,  -78],
                 [  26,  -92,  -79, ...,   28, -105,  -64],
                 [  44,  -74,  -92, ...,  -38, -172,  -80]]),
 'C4-A1': array([[ 554,  504,  478, ..., -226, -259, -238],
                 [-194, -226, -231, ...,    8,   41,   68],
                 [ 134,  164,  181, ..., -128, -188, -163],
                 ...,
                 [ -29,    4,    8, ...,    3,   35,    9],
                 [ -30,   -6,    0, ...,  -26,   -5,   -8],
                 [ -39,   -8,   -8, ...,  -46,  -36,  -53]])}
```

API Reference

openxdf.OpenXDF

Constructor

class `openxdf.OpenXDF(filepath=None, deidentify=True)`

Description

Wrapper for `.xdf` files with a number of attributes for accessing various pieces of information in the file.

Parameters

- Args:
  - filepath (*str*): File path to `.xdf` file.
  - deidentify (*bool*): (Default, True). Should the file be deidentified?

Example

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf
"<OpenXDF [Example]>
```

Attributes

Attribute	Description
id	Returns embedded patient ID.
start_time	Return the start time of the PSG study.
header	Returns general file encoding information.
sources	Return information on raw data sources.
montages	Return information on PSG montages (e.g. crossed sources that create a true EEG/EMG/EOG channel).
epochs	Return complex, interpreted information for each epoch (i.e. 30-second sleep period).
scoring	Return information about sleep scoring for each epoch (i.e. 30-second sleep period).
custom_event_list	Returns a dictionary of the custom events defined across scorers.
events	Returns a dict of all events across all scorers, including custom events.
dataframe	Returns DataFrame of scoring information and optional epoch and event information.

openxdf.OpenXDF.id

`OpenXDF.id`

Description

Returns embedded patient ID.

Parameters

- Returns:
  - *str*: patient ID.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.id
"Example"
```

openxdf.OpenXDF.start_time

`OpenXDF.start_time`

Description

Return the start time of the PSG study.

Parameters

- Returns:
  - *datetime*: Datetime object of PSG start time.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.start_time
datetime.datetime(2016, 4, 22, 22, 14, 57, 792999)
```

openxdf.OpenXDF.header

`OpenXDF.header`

Description

Returns general file encoding information.

Parameters

- Returns:
  - *dict*: Dictionary containing patient ID, epoch length, frame length, endianness, and `.xdf` file name.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.header
{"ID": "Example", "EpochLength": 30, "FrameLength": 1, "Endian": "little",
 "File": "example.data"}
```

openxdf.OpenXDF.sources

`OpenXDF.sources`

Description

Return information on raw data sources.

Parameters

- Returns:
  - *List[dict]*: List containin a dictionary of file information for each source in the `.xdf` file.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.sources
[{"SourceName": "FP1", "Unit": 1e-06, "UseGridScale": "false",
  "MinSamplingRate": 200, "MinSampleWidth": 1, "Ignore": "false",
  "PhysicalMax": 3199.9, "Signed": "true", "SampleWidth": 2,
  "SampleFrequency": 200, "DigitalMax": 32767, "DigitalMin": -32768,
  "PhysicalMin": -3200, "DigitalToVolts": 0.0976563},
  {...},
]
```

openxdf.OpenXDF.montages

`OpenXDF.montages`

Description

Return information on PSG montages (e.g. crossed sources that create a true EEG/EMG/EOG channel).

Parameters

- Returns:
  - *Dict[list]*: Dictionary of channels, where the value for each channel is a list that contains the two sources crossed to create that channel and the filtering information (e.g. "[Low, High]").

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.montages
{"O1-A2": [{"lead_1": "O1", "lead_2": "A2", "filter": ["0.531", "35.000"]}],
 "O2-A1": [{"lead_1": "O2", "lead_2": "A1", "filter": ["0.531", "35.000"]}],
 ...
}
```

openxdf.OpenXDF.epochs

`OpenXDF.epochs`

Description

Return complex, interpreted information for each epoch (i.e. 30-second sleep period).

Parameters

- Returns:
  - *List[dict]*: List of dictionaries, where each contains information for a single epoch in the sleep study. Each dictionary has information about the epoch number, body position, epoch number, number of microarousals, breath count, and many more variables.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.epochs
[{"Emg": 0, "EmgL": 76, "Eyes": 0, "REM": 1, "Alpha2": 0, "MyoL": 76,
  "EkgHL": 16694, "EkgAv": 60, "EkgEv": 0, "Oxy": 21328, "SaO2H": 129,
  "SaO2L": 5376, "SaO281to88": 129, "Cpap": 0, "Snore": 0, "Lad": 0,
  "Kcom": 0, "Body": 5, "Eye": 0, "MicS": 0, "MicA": 0, "EtCO2": 0, "AvgPh": 0,
  "BPres": 0, "Sigma2": 0, "SigmaPower": 0, "AlphaPower": 0, "ThetaPower": 0,
  "DeltaPower": 0, "BetaPower": 0, "BreathFreqTotal": 59.7186, "BreathCount": 3,
  "EmgHigh": 1, "EmgLow": 0, "StgStr": 10, "EpochNumber": 517, "Alpha1": 0,
  "Beta": 0, "Delta": 0, "Sigma": 0, "Theta": 0, "Artif": 0},
  ...,
]
```

openxdf.OpenXDF.scoring

`OpenXDF.scoring`

Description

Return information about sleep scoring for each epoch (i.e. 30-second sleep period).

Parameters

- Returns:
  - *List[dict]*: Returns a list containing a dictionary for each sleep technician/physician that scored the PSG file. Within a single scorer"s dictionary, there is the key"header", which contains the scorer"s name, and the key "staging", which is a list containing a dictionary for each sleep stage.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.scoring
[{"header": {"first_name": "Polysmith", "last_name": None},
  "staging": [{"EpochNumber": 517, "Stage": "R"},
              {"EpochNumber": 3, "Stage": "R"},
              {"EpochNumber": 520, "Stage": "R"},
              {"EpochNumber": 6, "Stage": "W"},
              ...
             ]
 },
...
]
```

openxdf.OpenXDF.custom_event_list

`OpenXDF.custom_event_list`

Description

Returns a dictionary of the custom events defined across scorers.

Parameters

- Returns:
  - *dict*: Dictionary of custom event definitions.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.custom_event_list
{"1": {"name": "Bruxism", "default_dur": 2, "min_dur": 1, "max_dur": 10},
 "2": {"name": "Cheyne-Stokes Breathing", "default_dur": 10, "min_dur": 1,
       "max_dur": 0},
 "3": {"name": "Microarousal", "default_dur": 3, "min_dur": 0, "max_dur": 0}
}
```

openxdf.OpenXDF.events

`OpenXDF.events`

Description

Returns a dict of all events across all scorers, including custom events.

Parameters

- Returns:
  - *dict*: Dictionary with key for each scorer. Within each scorer, there is a dictionary for each event type in the file.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.events
{"Polysmith": {
    "Hypopneas": [{"Time": "2016-04-22T22:16:42.632001000000002",
                   "Manual": "false", "Duration": "18.72", "Class": "obstructive",
                   "MinSaO2": "96"},
                  {"Time": "2016-04-22T22:17:11.711999100000000",
                   "Manual": "false", "Duration": "24.44", "Class": "obstructive",
                   "MinSaO2": "94"},
                   ...
                 ],
    "Desaturations": ...,
    ...
    },
...
}
```

openxdf.OpenXDF.dataframe

`OpenXDF.dataframe(epochs=True, events=True)`

Description

Returns DataFrame of scoring information and optional epoch and event information.

Parameters

- Args:
  - epochs (*bool*): [Default, True] Should the output include epoch information?
  - events (*bool*): [Default, True] Should the output include event information?
- Returns:
  - *pandas.DataFrame*: DataFrame arranged in "tidy" format.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> xdf.dataframe()
      EpochNumber Stage     Scorer  ...          Event      ElapsedTime
0               1  None     Andrea  ...  Desaturations  00:00:17.090004
1               1     W  Polysmith  ...            NaN              NaN
2               1  None       Wade  ...  Desaturations  00:00:17.094002
...
```

openxdf.Signal

Constructor

class `openxdf.Signal(xdf=None, filepath=None)`

Description

Wrapper for raw signal data files.

Parameters

- Args:
  - xdf (*openxdf.OpenXDF*): `OpenXDF` instance of related header file.
  - filepath (*str*): File path to raw signal file.

Example

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> signal = openxdf.Signal(xdf, "/path/to/file/.../example.data")
>>> signal
"<Signal [Example]>
```

Attributes

Attribute	Description
__frame__information	Returns information about the XDF dataframe and signal channels.
__source__information	Returns information about the XDF source channels.
list__channels	List all channels defined in XDF montage.
read__file	Read interlaced channels from binary signal file and return dictionary of bandpass-filtered signal data.
__edf__header	In progress.

openxdf.Signal.__frame__information

`Signal._frame_information`

Description

Returns information about the XDF dataframe and signal channels.

Parameters

- Returns:
  - *dict*: Dictionary containing Frame Length, Epoch Length, Endianness, Frame Width, and a list of information for each channel in the file.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> signal = openxdf.Signal(xdf, "/path/to/file/.../example.data")
>>> signal._frame_information
{'FrameLength': 1, 'EpochLength': 30, 'Endian': 'little', 'Num_Epochs': 924,
 'FrameWidth': 14750,
 'Channels': [
     {'SourceName': 'FP1', 'SampleWidth': 2, 'SampleFrequency': 200,
      'ChannelWidth': 400, 'Signed': 'true'},
     {'SourceName': 'FP2', 'SampleWidth': 2, 'SampleFrequency': 200,
      'ChannelWidth': 400, 'Signed': 'true'},
     ...
    ]
}
```

openxdf.Signal._source_information

`Signal._source_information`

Description

Returns information about the XDF source channels.

Parameters

- Returns:
  - *dict*: Dictionary containing the start location and width within a single frame for each channel.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> signal = openxdf.Signal(xdf, "/path/to/file/.../example.data")
>>> signal._source_information
{"PG1": {"Start": 8000, "Width": 400},
 "A2": {"Start": 9200, "Width": 400},
 ...,
}
```

openxdf.Signal.list_channels

`Signal.list_channels`

Description

List all channels defined in XDF montage.

Parameters

- Returns:
  - *list*: List containing all the channels in XDF montages.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> signal = openxdf.Signal(xdf, "/path/to/file/.../example.data")
>>> signal.list_channels
["EOG-L", "EOG-R", "F3-A2", ...]
```

openxdf.Signal.read_file

`Signal.read_file(channels=None)`

Description

Read interlaced channels from binary signal file and return dictionary of bandpass-filtered signal data.

Parameters

- Args:
    - channels (*list*): List of channel names (from `list_channels`).
- Returns:
    - *dict*: Dictionary of np.arrays, one per channel.

Examples

```
>>> xdf = openxdf.OpenXDF("/path/to/file/.../example.xdf")
>>> signal = openxdf.Signal(xdf, "/path/to/file/.../example.data")
>>> signal.read_file(["EOG-L", "C4-A1"])
{'EOG-L': array([[-890, -885, -803, ...,  393,  440,  422],
                 [ 494,  396,  451, ...,  323,  338,  420],
                 [ 504,  439,  493, ...,  251,  300,  244],
                 ...,
                 [  47, -104,  -79, ...,    9, -149,  -78],
                 [  26,  -92,  -79, ...,   28, -105,  -64],
                 [  44,  -74,  -92, ...,  -38, -172,  -80]]),
 'C4-A1': array([[ 554,  504,  478, ..., -226, -259, -238],
                 [-194, -226, -231, ...,    8,   41,   68],
                 [ 134,  164,  181, ..., -128, -188, -163],
                 ...,
                 [ -29,    4,    8, ...,    3,   35,    9],
                 [ -30,   -6,    0, ...,  -26,   -5,   -8],
                 [ -39,   -8,   -8, ...,  -46,  -36,  -53]])
}
```

openxdf.Signal._edf_header

`Signal._edf_header`

Description

In progress

Parameters

In progress

Examples

In progress