IPASC Data Conversion Tool

version

Data Acquisition and Management Theme of IPASC

September 25, 2020

Contents

Welcome to IPASC Data Conversion Tool’s documentation!

The photoacoustic data landscape is very heterogeneous, which can lead to problems in the exchange of photoacoustic images. Essentially, each vendor of photoacoustic devices has developed their own sophisticated data format that suffices their own needs.

While this is a very natural process it leads to two distinct problems:

1. Dependency on vendor-specific data access software to interpret recorded images.
2. Different definitions for similarly named fields

To this end, the Data Acquisition and Management thematic working group of IPASC has developed a standardised list of metadata parameters and published a corresponding document on their website in early 2020: **`IPASC Metadata definitions`**. This tool is based on the definitions contained in this document and is supposed to form the basis for a facilitated exchange of photoacoustic data.

Build the documentation

Run *sphinx-build -b pdf docs/source docs* in the top level folder to build the pdf documentation. The latest documentation will then be available under docs/ipasc_tool_documentation.pdf

Class references

Module: api

class `ipasc_tool.api.BaseAdapter.`**`BaseAdapter`**

abstract **`generate_binary_data`** **()** → numpy.ndarray
#TODO very detailed decription of how the binary meta data dump should be organized. :return: numpy array

`generate_meta_data` **()** → dict

Returns:

abstract **`generate_meta_data_device`** **()** → dict
# TODO this method can be implemented using the DeviceMetaDataCreator :return:

abstract **`set_metadata_value`** **(**metadata_tag: ipasc_tool.core.Metadata.MetaDatum**)** → object
This method must be implemented to yield appropriate data for all MetaDatum elements in the MetadataTags class.

Parameters: **metadata_tag** –

Returns:

class `ipasc_tool.api.adapters.DKFZ_CAMI_Experimental_System_Nrrd_File_Converter.`**`DKFZCAMIExperimentalSystemNrrdFileConverter`** (nrrd_file_path)

`generate_binary_data` **()** → numpy.ndarray
#TODO very detailed decription of how the binary meta data dump should be organized. :return: numpy array

`generate_meta_data_device` **()** → dict
# TODO this method can be implemented using the DeviceMetaDataCreator :return:

`set_metadata_value` **(**metadata_tag: ipasc_tool.core.Metadata.MetaDatum**)** → object
This method must be implemented to yield appropriate data for all MetaDatum elements in the MetadataTags class.

Parameters: **metadata_tag** –

Returns:

Module: core

class `ipasc_tool.core.Metadata.`**`EnumeratedString`** (tag, mandatory, dtype, unit='N/A', permissible_strings=None)

class `ipasc_tool.core.Metadata.`**`MetaDatum`** (tag: str, mandatory: bool, dtype: type, unit: str = 'N/A')
This class represents a meta datum. A meta datum contains all necessary information to fully characterize the meta information represented by an instance of this class.

class `ipasc_tool.core.Metadata.`**`MetadataAcquisitionTags`**
Binary time series data meta data tags

class `ipasc_tool.core.Metadata.`**`MetadataDeviceTags`**
This class defines the naming conventions of the

class `ipasc_tool.core.Metadata.`**`NDimensionalNumpyArray`** (tag, mandatory, dtype, unit='N/A', expected_array_dimension=1)

class `ipasc_tool.core.Metadata.`**`NonNegativeNumber`** (tag, mandatory, dtype, unit='N/A')

class `ipasc_tool.core.Metadata.`**`NonNegativeNumbersInArray`** (tag, mandatory, dtype, unit='N/A')

class `ipasc_tool.core.Metadata.`**`NonNegativeWholeNumber`** (tag, mandatory, dtype, unit='N/A')

class `ipasc_tool.core.Metadata.`**`NumberWithUpperAndLowerLimit`** (tag, mandatory, dtype, unit='N/A', lower_limit=- inf, upper_limit=inf)

class `ipasc_tool.core.Metadata.`**`UnconstrainedMetaDatum`** (tag, mandatory, dtype, unit='N/A')

class `ipasc_tool.core.PAData.`**`PAData`** (binary_time_series_data: numpy.ndarray, meta_data_acquisition: dict = None, meta_data_device: dict = None)
The PAData class is the core class for accessing the information contained in the HDF5 files. Using the iohandler.file_reader.load_data method yields an instance of this class.
It is structured into three main parts: (1) a numpy array containing the binary data (2) a dictionary with the acquisition metadata (3) a dictionary with the device meta data
Furthermore, this class contains convenience methods to access all fields within the HDF5 dictionary, without the necessity to know the internal structure by heart.

`get_acquisition_meta_datum` **(**meta_data_tag: ipasc_tool.core.Metadata.MetaDatum**)** → object
This method returns data from the acquisition meta data dictionary

Parameters: **meta_data_tag** – the MetaDatum instance for which to get the information.

Returns: return value might be None, if the specified meta data tag was not found in the dictionary.

`get_angular_response` **(**identifier=None**)**
The angular response field characterizes the angular sensitivity of the detection element to the incident angle (relative to the elements orientation) of the incoming pressure wave.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_beam_divergence` **(**identifier=None**)**
The beam divergence angles represent the opening angles of the laser beam from the illuminator shape with respect to the orientation vector. This angle represented by the standard deviation of the beam divergence.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_beam_profile` **(**identifier=None**)**
The beam intensity profile is a function of a spatial position that specifies the relative laser beam intensity according to the planar emitting surface of the illuminator shape.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_compression` **()**
The compression field is representative of the compression method that was used to compress the binary data. E.g. one of ‘raw’, ‘gzip’, …

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_coupling_agent` **()**
A string representation of the acoustic coupling agent that was used. For example, the following options are possible: D2O, H2O and US-gel.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_custom_meta_datum` **(**meta_data_tag: str**)** → object
This method returns data from the acquisition meta data dictionary.

Parameters: **meta_data_tag** – a string instance for which to get the information.

Returns: return value might be None, if the specified meta data tag was not found in the dictionary.

`get_data_UUID` **()**
128-bit Integer displayed as a hexadecimal string in 5 groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters. The UUID is randomly generated using the UUID Version 4 standard.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_data_type` **()**
The data type field represents the datatype of the binary data. This field is given in the C++ data type naming convention. E.g. ‘short’, ‘unsigned short’, ‘int’, ‘unsigned int’, ‘long’, ‘unsigned long’, ‘long long’, ‘float’, ‘double’, ‘long double’.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_detector_ids` **()**

Returns: a list of all ids of the detection elements

`get_detector_orientation` **(**identifier=None**)**
The element orientation defines the rotation of the detection element in 3D cartesian coordinates [r1, r2, r3] in radians.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_detector_position` **(**identifier=None**)**
The element position defines the position of the detection element centroid in 3D cartesian coordinates [x1, x2, x3].

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_detector_size` **(**identifier=None**)**
The element size defines the size of the detection element in 3D cartesian coordinates [x1, x2, x3] relative to its position and orientation.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_device_reference` **()**
A reference to the UUID of the PA imaging device description as defined in part 1.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_device_uuid` ()
The UUID is a universally unique identifier to the device description that can be referenced. :return: return value can be None, of no UUID was found in the meta data.

`get_dimensionality` ()
The dimensionality field represents the acquisition format of the binary data and specifies the number of spatiotemporal dimensions of the data that is comprised of one or more frames. E.g. ‘1D’, ‘2D’, ‘3D’, ‘1D+t’, 2D+t’, ‘3D+t’. In this notion, the time series sampling of one transducer would count as a “spatial” dimension. These are defined as 1D = [■], 2D = [x1, ■], 3D = [x1, ■, x2]. The “+t” will then add a time dimension for multiple of these frames.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_element_dependent_gain` ()
The element-dependent gain field is a 2D array that contains the relative factors which have been used to perform apodization.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_encoding` ()
The encoding field is representative of the character set that was used to encode the binary data and the metadata. E.g. one of ‘UTF-8’, ‘ASCII’, ‘CP-1252’, …

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_energy_profile` (identifier=None**)**
The laser energy profile field is a discretized functional of wavelength (nm) that represents the laser energy of the illuminator with regard to the wavelength. Thereby, systematic differences in multispectral image acquisitions can be accounted for.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_field_of_view` ()
The field of view defines an approximate cube of the area detectable by the PA imaging device in 3D cartesian coordinates [x1, x2, x3]. The field of view always starts in the origin of the coordinate system (which is defined as the centroid of the top-left transducer element when looking at the device normal to the imaging plane) and expands in the positive x1, x2, x3 directions.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_frequency_filter` ()

The frequency threshold levels that have been applied to filter the raw time series data.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_frequency_response` (identifier=None**)**
The frequency response is a functional that characterizes the response of the detection element to the frequency of the incident pressure waves.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_illuminator_ids` () → list

Returns: a list of all ids of the illumination elements

`get_illuminator_orientation` (identifier=None**)**
The illuminator orientation defines the rotation of the illuminator in 3D cartesian coordinates [r1, r2, r3]. It is the normal of the planar illuminator surface.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_illuminator_position` (identifier=None**)**

The illuminator position defines the position of the illuminator centroid in 3D cartesian coordinates [x1, x2, x3] .

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_illuminator_size` **(**identifier=None**)**
The illuminator shape defines the shape of the optical fibres, so it describes whether the illuminator is a point illuminator, or has a more continuous form. Illuminators can only have planar emitting surfaces.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_number_of_detectors` **()**
The number of detectors quantifies the number of transducer elements that are used in the respective PA imaging device. Each of these transducer elements is described by a set of detection geometry parameters.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_number_of_illuminators` **()**
The number of illuminators quantifies the number of illuminators that are used in the respective PA imaging device. Each of these illuminators is described by a set of illumination geometry parameters.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_overall_gain` **()**
The overall gain is a single value describing a factor that has been applied to all values of the raw time series data.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_pulse_laser_energy` **()**
The pulse laser energy field specifies the pulse-to-pulse laser energy that was measured for the acquisition of the raw time series data.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_pulse_width` **(**identifier=None**)**
The pulse duration or pulse width describes the total length of a laser pulse, measured as the time interval between the half-power points on the leading and trailing edges of the pulse.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_sampling_rate` **()**
The A/D sampling rate refers to the rate at which samples of the analog signal are taken to be converted into digital form.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_scanning_method` **()**
A string representation of the scanning method that was used.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_sizes` **()**
The sizes field quantifies the number of data points in each of the dimensions specified in the dimensionality field. e.g. [128, 2560, 26] with a “2D+t” dimensionality.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_stability_profile` **(**identifier=None**)**
The laser noise profile field is a functional of wavelength (nm) that represents the standard deviation of the pulse-to-pulse laser energy of the illuminator with regard to the wavelength.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_temperature` ()
The temperature control field indicates the temperature during image acquisition.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_time_gain_compensation` ()
The time gain compensation field is a 1D array that contains the relative factors which have been used to modify the time series data to correct for the effect of attenuation.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_time_stamps` ()
The frame acquisition timestamps field indicates the timestamp of the acquisition system.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_wavelength_range` (identifier=None**)**
The wavelength range quantifies the wavelength range that the illuminator is capable of generating by reporting three values: the minimum wavelength max, the maximum wavelength max and a metric for the accuracy accuracy: (min, max, accuracy). This parameter could for instance be (700, 900, 1.2), meaning that this illuminator can be tuned from 700 nm to 900 nm with an accuracy of 1.2 nm.

Returns: return value can be None, of the key was not found in the meta data dictionary.

`get_wavelengths` ()
The acquisition optical wavelengths field is a 1D array that contains a list of all wavelengths used for the image acquisition.

Returns: return value can be None, of the key was not found in the meta data dictionary.

Module: iohandler

`ipasc_tool.iohandler.file_reader.`**`load_data`** (path: str)
TODO :param path: Path to an hdf5 file containing PAData. :return: PAData instance

`ipasc_tool.iohandler.file_writer.`**`write_data`** (path: str, pa_data: ipasc_tool.core.PAData.PAData)
TODO :param path: Path to save an hdf5 file containing PAData. :param pa_data: PAData instance :return:

Module: qualitycontrol

`ipasc_tool.qualitycontrol.CompletenessChecker.`**`check_metadatum_from_dict`** (dictionary: dict, metadatum: ipasc_tool.core.Metadata.MetaDatum)

Parameters:
- **dictionary** –
- **meta_datum** –

Returns: [log, count]

Index

Python Module Index