McsPyDataTools Documentation

Release 0.3.0

Multi Channel Systems MCS GmbH

Jun 12, 2018

CONTENTS

The aim of the McsPyDataTools package is to provide a convenient python interface to access the content of HDF5 data files created by the Multi Channel DataManager (Multi Channel Systems MCS GmbH).

Contents:

CHAPTER

ONE

MCSPYDATATOOLS API REFERENCE

1.1 McsPy

McsPy is a Python module/package to read, handle and operate on HDF5-based raw data files converted from recordings of devices of the Multi Channel Systems MCS GmbH.

copyright

3. 2018 by Multi Channel Systems MCS GmbH

license see LICENSE for more details

class McsPy.**McsHdf5Protocols**

Class of supported MCS-HDF5 protocol types and version ranges

Entry: (Protocol Type Name => Tuple of supported version range from (including) the first version entry up to (including) the second version entry)

classmethod check_protocol_type_version(*protocol_type_name*, *version*)

Check if the given version of a protocol is supported by the implementation

Parameters

- **protocol_type_name** – name of the protocol that is tested
- **version** – version number that should be checked

Returns is true if the given protocol and version is supported

class McsPy.**McsHdf5Types**

Class of supported MCS-HDF5 file structure types and version ranges

Entry: (Protocol TypeID => Tuple of supported version range from (including) the first version entry up to (including) the second version entry)

classmethod check_type_version(*typeID*, *version*)

Check if the given version of a type is supported by the implementation

Parameters

- **protocol_type_name** – name of the protocol that is tested
- **version** – version number that should be checked

Returns is true if the given type and version is supported

classmethod get_mcs_class_name(*typeID*, *version=None*)

Returns the McsPy class name, that corresponds to a given Mcs HDF5 file structure type. The function also checks if the requested class supports the Mcs HDF5 file structure type version

Parameters

- **typeID** – name of the type that is tested
- **version** – version number that should be checked

Returns a McsCMOSMEA class if the given type and version is supported

1.2 The ‘’McsData” module

class McsPy.McsData.**RawData**(*raw_data_path*)
This class holds the information of a complete MCS raw data file

recordings
Access recordings

class McsPy.McsData.**Recording**(*recording_grp*)
Container class for one recording

Provides the content of the HDF5 *Folder “Recording_x”* in Python.

analog_streams
Access all analog streams - collection of *AnalogStream* objects

frame_streams
Access all frame streams - collection of *FrameStream* objects

event_streams
Access event streams - collection of *EventStream* objects

segment_streams
Access segment streams - - collection of SegementStream objects

timestamp_streams
Access timestamp streams - collection of TimestampStream objects

duration_time
Duration of the recording

1.2.1 *Data-Stream*-Structures containing the data

class McsPy.McsData.**Stream**(*stream_grp*, *info_type_name=None*)
Base class for all stream types

class McsPy.McsData.**AnalogStream**(*stream_grp*)
Container class for one analog stream of several channels. Description for each channel is provided by a channel-associated object of *ChannelInfo*

Provides the content of the HDF5 *Sub-folder “Stream_x” of “AnalogStream”* in Python.

get_channel_in_range(*channel_id*, *idx_start*, *idx_end*)
Get the signal of the given channel over the curse of time and in its measured range.

Parameters

- **channel_id** – ID of the channel
- **idx_start** – index of the first sampled signal value that should be returned (0 <= idx_start < idx_end <= count samples)
- **idx_end** – index of the last sampled signal value that should be returned (0 <= idx_start < idx_end <= count samples)

Returns Tuple (vector of the signal, unit of the values)

get_channel_sample_timestamps(*channel_id*, *idx_start*, *idx_end*)
Get the timestamps of the sampled values.

Parameters

- **channel_id** – ID of the channel
- **idx_start** – index of the first signal timestamp that should be returned (0 <= idx_start < idx_end <= count samples)
- **idx_end** – index of the last signal timestamp that should be returned (0 <= idx_start < idx_end <= count samples)

Returns Tuple (vector of the timestamps, unit of the timestamps)

class McsPy.McsData.**FrameStream**(*stream_grp*)
Container class for one frame stream with different entities

Provides the content of the HDF5 *Subfolder "Stream_x" of "FrameStream"* in Python.

class McsPy.McsData.**FrameEntity**(*frame_entity_group*, *frame_info*)
Contains the stream of a specific frame entity. Meta-Information for this entity is available via an associated object of `FrameEntityInfo`

Provides the content of the HDF5 *Subfolder "Stream_x" of "FrameStream"* and *Subfolder "FrameDataEntity_x"* in Python.

get_sensor_signal(*sensor_x*, *sensor_y*, *idx_start*, *idx_end*)
Get the signal of a single sensor over the curse of time and in its measured range.

Parameters

- **sensor_x** – x coordinate of the sensor
- **sensor_y** – y coordinate of the sensor
- **idx_start** – index of the first sampled frame that should be returned (0 <= idx_start < idx_end <= count frames)
- **idx_end** – index of the last sampled frame that should be returned (0 <= idx_start < idx_end <= count frames)

Returns Tuple (vector of the signal, unit of the values)

get_frame_timestamps(*idx_start*, *idx_end*)
Get the timestamps of the sampled frames.

Parameters

- **idx_start** – index of the first sampled frame that should be returned (0 <= idx_start < idx_end <= count frames)
- **idx_end** – index of the last sampled frame that should be returned (0 <= idx_start < idx_end <= count frames)

Returns Tuple (vector of the timestamps, unit of the timestamps)

class McsPy.McsData.**EventStream**(*stream_grp*)
Container class for one event stream with different entities

Provides the content of the HDF5 *Subfolder "Stream_x" of "EventStream"* in Python.

class McsPy.McsData.**EventEntity**(*event_data*, *event_info*)
Contains the event data of a specific entity. Meta-Information for this entity is available via an associated object of *EventEntityInfo*

Maps data event entity content of the HDF5 *Subfolder "Stream_x" of "EventStream"* to Python structures.

count
Number of contained events

get_events(*idx_start=None*, *idx_end=None*)
Get all n events of this entity of the given index range (idx_start <= idx < idx_end)

Parameters

- **idx_start** – start index of the range (including), if nothing is given -> 0
- **idx_end** – end index of the range (excluding, if nothing is given -> last index

Returns Tuple of (2 x n matrix of timestamp (1. row) and duration (2. row), Used unit of time)

get_event_timestamps(*idx_start=None*, *idx_end=None*)
Get all n event timestamps of this entity of the given index range

Parameters

- **idx_start** – start index of the range, if nothing is given -> 0
- **idx_end** – end index of the range, if nothing is given -> last index

Returns Tuple of (n-length array of timestamps, Used unit of time)

get_event_durations(*idx_start=None*, *idx_end=None*)
Get all n event durations of this entity of the given index range

Parameters

- **idx_start** – start index of the range, if nothing is given -> 0
- **idx_end** – end index of the range, if nothing is given -> last index

Returns Tuple of (n-length array of duration, Used unit of time)

class McsPy.McsData.**SegmentStream**(*stream_grp*)
Container class for one segment stream of different segment entities

Provides the content of the HDF5 *Subfolder "Stream_x" of "SegmentStream"* in Python.

class McsPy.McsData.**SegmentEntity**(*segment_data*, *segment_ts*, *segment_info*)
Segment entity class, Meta-Information for this entity is available via an associated object of *SegmentEntityInfo*

DataSybType != Average → Maps segement entity content of the HDF5 *Subfolder "Stream_x" of "SegmentStream"* to Python structures.

segment_sample_count
Number of contained samples of segments (2d) or multi-segments (3d)

segment_count
Number of segments that are sampled for one time point (2d) -> 1 and (3d) -> n

get_segment_in_range(*segment_id*, *flat=False*, *idx_start=None*, *idx_end=None*)
Get the a/the segment signals in its measured range.

Parameters

- **segment_id** – id resp. number of the segment (0 if only one segment is present or the index inside the multi-segment collection)

- **flat** – true -> one-dimensional vector of the sequentially ordered segments, false -> k x n matrix of the n segments of k sample points
- **idx_start** – index of the first segment that should be returned (0 <= idx_start < idx_end <= count segments)
- **idx_end** – index of the last segment that should be returned (0 <= idx_start < idx_end <= count segments)

Returns Tuple (of a flat vector of the sequentially ordered segments or a k x n matrix of the n segments of k sample points depending on the value of *flat* , and the unit of the values)

get_segment_sample_timestamps(*segment_id*, *flat=False*, *idx_start=None*, *idx_end=None*)
Get the timestamps of the sample points of the measured segment.

Parameters

- **segment_id** – id resp. number of the segment (0 if only one segment is present or the index inside the multi-segment collection)
- **flat** – true -> one-dimensional vector of the sequentially ordered segment timestamps, false -> k x n matrix of the k timestamps of n segments
- **idx_start** – index of the first segment for that timestamps should be returned (0 <= idx_start < idx_end <= count segments)
- **idx_end** – index of the last segment for that timestamps should be returned (0 <= idx_start < idx_end <= count segments)

Returns Tuple (of a flat vector of the sequentially ordered segments or a k x n matrix of the n segments of k sample points depending on the value of *flat* , and the unit of the values)

class McsPy.McsData.**AverageSegmentTuple**(*mean*, *std_dev*, *time_tick_unit*, *signal_unit*)
Named tuple that describe one or more average segments (mean, std_dev, time_tick_unit, signal_unit).

Note:

- *mean* - mean signal values
- *std_dev* - standard deviation of the signal value (it is 0 if there was only one sample segment)
- *time_tick_unit* - sampling interval with time unit
- *signal_unit* - measured unit of the signal

---

mean
Alias for field number 0

signal_unit
Alias for field number 3

std_dev
Alias for field number 1

time_tick_unit
Alias for field number 2

class McsPy.McsData.**AverageSegmentEntity**(*segment_average_data*, *segment_average_annotation*, *segment_info*)
Contains a number of signal segments that are calcualted as averages of number of segments occured in a given time range. Meta-Information for this entity is available via an associated object of *SegmentEntityInfo*

DataSybType == Average → Maps segment entity content of the HDF5 *DataSubType-Average: Subfolder "Stream_x" of "SegmentStream"* to Python structures.

number_of_averages
Number of average segments inside this average entity

sample_length
Number of sample points of an average segment

time_ranges()
List of time range tuples for all contained average segments

Returns List of tuple with start and end time point

time_range(*average_segment_idx*)
Get the time range for that the average segment was calculated

Parameters **average_segment_idx** – index resp. number of the average segment

Returns Tuple with start and end time point

average_counts()
List of counts of samples for all contained average segments

Parameters **average_segment_idx** – id resp. number of the average segment

Returns sample count

average_count(*average_segment_idx*)
Count of samples that were used to calculate the average

Parameters **average_segment_idx** – id resp. number of the average segment

Returns sample count

get_scaled_average_segments()
Get all contained average segments in its measured physical range.

Returns *AverageSegmentTuple* containing the k x n matrices for mean and standard deviation of all contained average segments n with the associated sampling and measuring information

get_scaled_average_segment(*average_segment_idx*)
Get the selected average segment in its measured physical range.

Parameters **segment_idx** – index resp. number of the average segment

Returns *AverageSegmentTuple* containing the mean and standard deviation vector of the average segment with the associated sampling and measuring information

get_average_segments()
Get all contained average segments AD-offset in ADC values with its measuring conditions

Returns *AverageSegmentTuple* containing the mean and standard deviation vector of the average segment in ADC steps with sampling tick and ADC-Step definition

get_average_segment(*average_segment_idx*)
Get the AD-offset corrected average segment in ADC values with its measuring conditions

Parameters **segment_id** – id resp. number of the segment

Returns *AverageSegmentTuple* containing the k x n matrices for mean and standard deviation of all contained average segments in ADC steps with sampling tick and ADC-Step definition

class McsPy.McsData.**TimeStampStream**(*stream_grp*)
Container class for one timestamp stream with different entities

Provides the content of the HDF5 *Subfolder "Stream_x" of "TimeStampStream"* in Python.

class McsPy.McsData.**TimeStampEntity**(*timestamp_data*, *timestamp_info*)
Time-Stamp entity class, Meta-Information for this entity is available via an associated object of `TimestampEntityInfo`

Maps data timestamp entity data of the HDF5 *Subfolder "Stream_x" of "TimeStampStream"* to Python structures.

count
Number of contained timestamps

get_timestamps(*idx_start=None*, *idx_end=None*)
Get all n time stamps of this entity of the given index range (idx_start <= idx < idx_end)

Parameters

- **idx_start** – start index of the range (including), if nothing is given -> 0
- **idx_end** – end index of the range (excluding, if nothing is given -> last index

Returns Tuple of (n-length array of timestamps, Used unit of time)

1.2.2 *Info*-Classes containing Meta-Information for the data

class McsPy.McsData.**Info**(*info_data*)
Base class of all info classes

Derived classes contain meta information for data structures and fields.

get_field(*name*)
Get the field with that name -> access to the raw info array

group_id
Get the id of the group that the objects belongs to

label
Label of this object

data_type
Raw data type of this object

class McsPy.McsData.**ChannelInfo**(*info_version*, *info*)
Contains all describing meta data for one sampled channel

channel_id
Get the ID of the channel

row_index
Get the index of the row that contains the associated channel data inside the data matrix

adc_step
Size and unit of one ADC step for this channel

version
Version number of the Type-Definition

class McsPy.McsData.**InfoSampledData**(*info*)
Base class of all info classes for evenly sampled data

sampling_frequency
Get the used sampling frequency in Hz

sampling_tick
Get the used sampling tick

class McsPy.McsData.**EventEntityInfo**(*info_version*, *info*)
Contains all meta data for one event entity

id
Event ID

raw_data_bytes
Lenght of raw data in bytes

source_channel_ids
ID's of all channels that were involved in the event generation.

source_channel_labels
Labels of the channels that were involved in the event generation.

version
Version number of the Type-Definition

class McsPy.McsData.**SegmentEntityInfo**(*info_version*, *info*, *source_channel_infos*)
Contains all meta data for one segment entity

id
Segment ID

pre_interval
Interval [start of the segment <- defining event timestamp]

post_interval
Interval [defining event timestamp -> end of the segment]

type
Type of the segment like 'Average' or 'Cutout'

count
Count of segments inside the segment entity

version
Version number of the Type-Definition

class McsPy.McsData.**TimeStampEntityInfo**(*info_version*, *info*)
Contains all meta data for one timestamp entity

id
Timestamp entity ID

unit
Unit in which the timestamps are measured

exponent
Exponent for the unit in which the timestamps are measured

measuring_unit
Unit in which the timestamp entity was measured

data_type
DataType for the timestamps

source_channel_ids
ID's of all channels that were involved in the timestamp generation.

source_channel_labels
Labels of the channels that were involved in the timestamp generation.

version
Version number of the Type-Definition

1.3 The ''McsCMOS" module

1.3.1 McsCMOS

Wrapper and Helper to access MCS CMOS Data within H5 Files

copyright

3. 2018 by Multi Channel Systems MCS GmbH

license see LICENSE for more details

class McsPy.McsCMOS.**CMOSConvProxy**(*parent*)
Private Class, should be embedded within a CMOSData Object. A proxy that transparently converts raw data to calibrated data.

shape
Shape of the data

class McsPy.McsCMOS.**CMOSData**(*path*)
Wrapper for a HDF5 File containing CMOS Data

class McsPy.McsCMOS.**CMOSSpikes**(*path*)
Wrapper for a HDF5 File containing CMOS Spike Data. Spike Information is accessible through the .spike Member, Waveform Information (if available) through the .waveforms Member.

CHAPTER

TWO

MCS HDF5 FORMAT DEFINITIONS

2.1 Definition of the HDF5 format for raw data

MCS-HDF5 Protocol Type: RawData (Raw-Data protocol)

Protocol Version: 3 based on the definitions of RawDataFileIO in version 10.

All strings are only ASCII-encoded

2.1.1 Changelog

Version 1:

- Initial draft

Version 2:

- New Root-Folder attributes added to detect name and version of the creating application and library

Version 3:

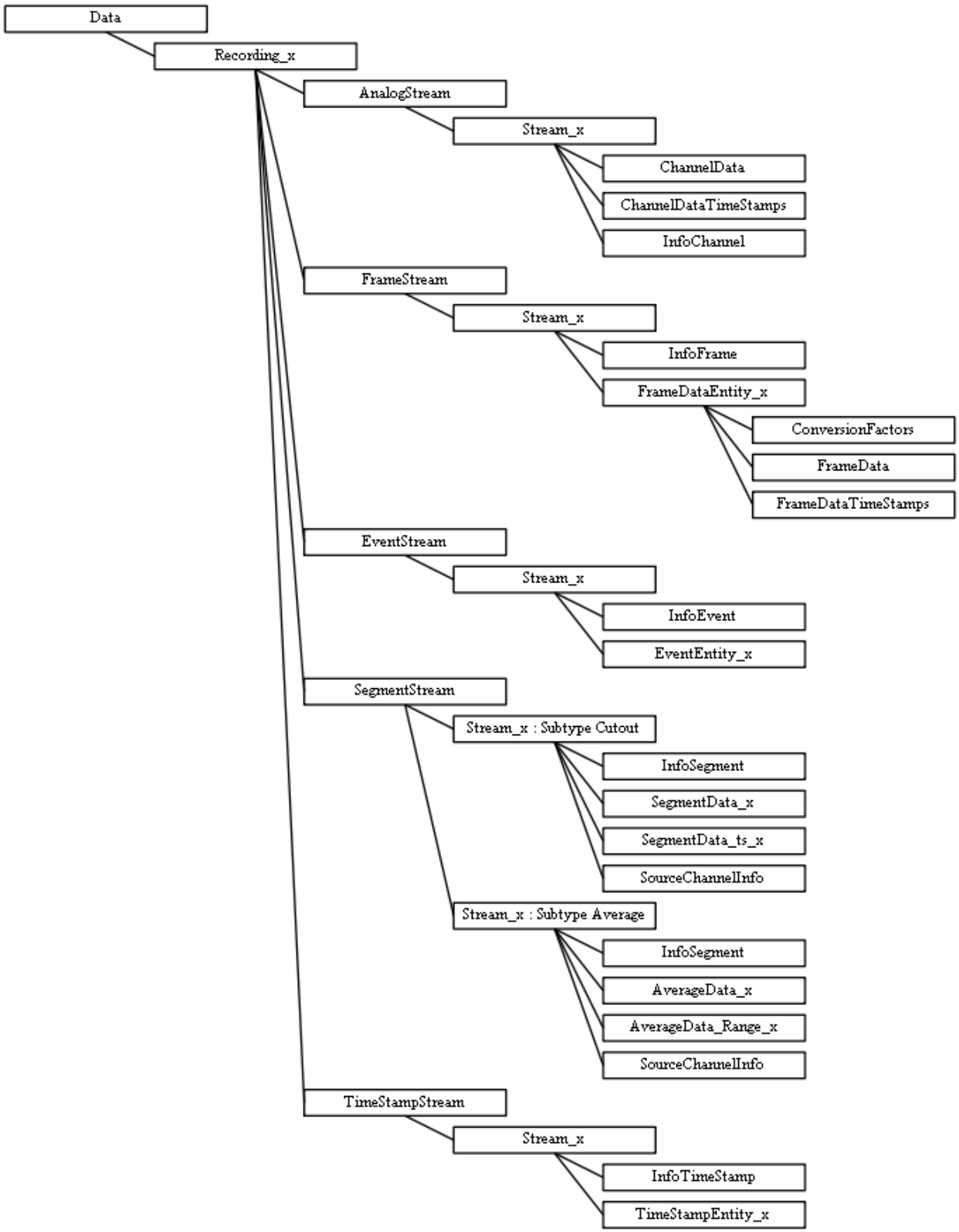
- Data structures for **DataSubType::Average** of **StreamType::Segment** added

Hierarchy

Root-Folder "/"

Contains all information for one experiment - measured data (inside the folder Data) and a description (possibly in the future) inside the folder Experiment/Description/...

Attributes:


Data
Recording_x
AnalogStream
Stream_x
ChannelData
ChannelDataTimeStamps
InfoChannel
FrameStream
Stream_x
InfoFrame
FrameDataEntity_x
ConversionFactors
FrameData
FrameDataTimeStamps
EventStream
Stream_x
InfoEvent
EventEntity_x
SegmentStream
Stream_x : Subtype Cutout
InfoSegment
SegmentData_x
SegmentData_ts_x
SourceChannelInfo
Stream_x : Subtype Average
InfoSegment
AverageData_x
AverageData_Range_x
SourceChannelInfo
TimeStampStream
Stream_x
InfoTimeStamp
TimeStampEntity_x

Name	Description	Data Type	MCS-HDF5 Protocol Version
McsHdf5ProtocolType	Type of the used MCS-HDF5 protocol definition (e.g. **RawData** for the raw data MCS-HDF5 definitions)	[**String**,**Scalar**]	1 ≤
McsHdf5ProtocolVersion	Version number of the used MCS-HDF5 protocol	[**Integer**,**Scalar**]	1 ≤
GeneratingApplicationName	Name of the application that generated this HDF5 file	[**String**,**Scalar**]	2 ≤
GeneratingApplicationVersion	Version of the application that generated this HDF5 file	[**String**,**Scalar**]	2 ≤
McsDataToolsVersion	Version of the McsDataTools library that was used by the application to create the HDF5 file	[**String**,**Scalar**]	2 ≤

Datasets:

- none

Folder "Data"

Navigation: /Data

Contains all recordings for this experiment.

Attributes:

Name	Description	Data Type
ProgramName	Name of the recording program	[**String**,**Scalar**]
ProgramVersion	Version number of the recording program	[**String**,**Scalar**]
MeaName	Name of the recorded MEA	[**String**,**Scalar**]
MeaLayout	Layout descriptor	[**String**,**Scalar**]
MeaSN	Serial number of the MEA	[**String**,**Scalar**]
Date	Date of the recording	[**String**,**Scalar**]
DateInTicks	Date of the recording in .NET ticks (100 ns)	[**Long**(64-bit Integer),**Scalar**]
FileGUID	GUID of the converted raw data file	[**String**,**Scalar**]
Comment	Comment	[**String**,**Scalar**]

Datasets:

- none

Folder "Recording_x"

Navigation: /Data/Recording_x

Contains all recorded streams for recording x.

Attributes:

Name	Description	Data Type
RecordingID	Recording ID	[**Integer**(32-bit Integer),**Scalar**]
RecordingType	Recording type	[**String**,**Scalar**]
TimeStamp	Start time of the recording in microseconds	[**Long**(64-bit Integer),**Scalar**]
Duration	Total recording duration in microseconds (This duration can differ from the actual duration of the recorded data!!!)	[**Long**(64-bit Integer),**Scalar**]
Label	Label	[**String**,**Scalar**]
Comment	Comment	[**String**,**Scalar**]

Datasets:

- none

Folder "AnalogStream"

Navigation: /Data/Recording_x/AnalogStream

(Organisational) folder for all channel-based streams of this recording

Attributes:

- none

Datasets:

- none

Sub-folder "Stream_x" of "AnalogStream"

Navigation: /Data/Recording_x/AnalogStream/Stream_x

Container for an analog stream

Attributes:

Name	Description	Data Type	StreamInfoVersion
StreamInfoVersion	Version number of the meta information structure	[**Int**(32-bit Integer),**Scalar**]	$1 \leq$
Label	Label	[**String**,**Scalar**]	$1 \leq$
SourceStreamGUID	GUID of the source streams	[**String**,**Scalar**]	$1 \leq$
StreamGUID	GUID	[**String**,**Scalar**]	$1 \leq$
StreamType	Type of the stream, e.g. **Electrode**	[**String**,**Scalar**]	$1 \leq$
DataSubType	Sub-type of the analog stream (e.g. **Analog**)	[**String**,**Scalar**]	$1 \leq$

Datasets:

- Matrix InfoChannel $\rightarrow$ n $\times$ 16 matrix of describing information vectors for the n channels:
  - Attributes: InfoVersion $\rightarrow$ Version number of the Info-Objects [**Int**(32-bit Integer),**Scalar**]

Name	Description	Data Type	InfoVersion
ChannelID	ID of the channel as given by the recording software	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
RowIndex	Row number of this channel inside the **ChannelData** matrix where the data of this channel is stored	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
GroupID	ID of the group that this channel belongs to	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
Label	Label of the channel	[**String**,**Array**]	1 ≤
RawDataType	Type of the raw data	[**String**,**Array**]	1 ≤
Unit	Physical unit of the measured sensor value	[**String**,**Array**]	1 ≤
Exponent	Exponent n ⇒ 1En resp. 10^n in which the channel values magnitude is measured (e.g. k,m,µ,...)	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
ADZero	ADC-Step that represents the 0-point of the measuring range of the ADC	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
Tick	Sample tick Δ between two sample points of a channel in µs ⇒ sampling frequency = 1000000 / Δ	[**Long**(64-bit Integer),**Array**(Size 1)]	1 ≤
ConversionFactor	Conversion factor for the mapping ADC-Step ⇒ measured value	[**Long**(64-bit Integer),**Array**(Size 1)]	1 ≤
ADCBits	Number of bits used by the AD-Converter	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
HighPassFilterType	Type of the high-pass filter (empty string if not available)	[**String**,**Scalar**]	1 ≤
HighPassFilterCutOffFrequency	Cut-off frequency of the high-pass filter ('-1'-String if not available)	[**String**,**Scalar**]	1 ≤
HighPassFilterOrder	Order of the high-pass filter (-1 if not available)	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
LowPassFilterType	Type of the low-pass filter (empty string if not available)	[**String**,**Scalar**]	1 ≤
LowPassFilterCutOffFrequency	Cut-off frequency of the low-pass filter ('-1'-String if not available)	[**String**,**Scalar**]	1 ≤
LowPassFilterOrder	Order of the low-pass filter (-1 if not available)	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤

- 2-dimensional Data-Matrix ChannelData → Data for sampled channels organized as n × m matrix ⇒ one row per channel and one column per sample time point
  - reconstruct the value of the measured signal: $y(\text{channel}, t_{ind})$ = (ChannelData[InfoChannel[channel].RowIndex, t_{ind}] − ADZero) ∗ InfoChannel[channel].ConversionFactor ∗ $10^{\text{InfoChannel[channel].Exponent}}$ in InfoChannel[channel].Unit
  - reconstruct the sample time point: $t = t_{ind} *$ InfoChannel[channel].Tick in μs
- Matrix ChannelDataTimeStamps → k × 3 matrix of segments where the rows are one segment and the columns are:
  - first column → time stamp of the first sample point of the segment
  - second column → first index (column) of the segment in **ChannelData**
  - third column → last index (column) of the segment in **ChannelData**

Folder "FrameStream"

Navigation: /Data/Recording_x/FrameStream

(Organisational) folder for all frame-based streams of this recording

Attributes:

- none

Datasets:

- none

Subfolder "Stream_x" of "FrameStream"

Navigation: /Data/Recording_x/FrameStream/Stream_x

Folder that contains all Frame-Entities of one Frame-Stream:

Attributes:

Name	Description	Data Type	StreamInfoVersion
StreamInfoVersion	Version number of the meta information structure	[**Int**(32-bit Integer),**Scalar**]	1 ≤
Label	Label	[**String**,**Scalar**]	1 ≤
SourceStreamGUID	GUID of the source stream	[**String**,**Scalar**]	1 ≤
StreamGUID	GUID	[**String**,**Scalar**]	1 ≤
StreamType	Type of the stream **Frame**	[**String**,**Scalar**]	1 ≤
DataSubType	Sub-type of the event stream (e.g. **SpikeTimeStamp**)	[**String**,**Scalar**]	1 ≤

Datasets:

- Matrix InfoFrame → n × 24 matrix of describing information vectors for the n Frame-Entities:
  - Attributes: InfoVersion → Version number of the Info-Objects [**Int**(32-bit Integer),**Scalar**]

Name	Description	Data Type	InfoVersion
FrameID	ID of the frame entity as given by the recording software	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
FrameDataID	ID of the frame entity inside the stream folder that maps this information vector to the entity folder (**FrameDataID** → subfolder **FrameDataEntity_FrameDataID**)	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
GroupID	ID of the group that this frame entity belongs to	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
Label	Label of the entity	[**String**,**Array**]	1 ≤
RawDataType	Type of the raw data	[**String**,**Array**]	1 ≤
Unit	Physical unit of the measured sensor value	[**String**,**Array**]	1 ≤
Exponent	Exponent n ⇒ 1En resp. 10^n in which the sensor values magnitude is measured (e.g. k,m,µ,. . .)	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
ADZero	ADC-Step that represents the 0-point of the measuring range of the ADC	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
ADCBits	Number of bits used by the AD-Converter	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
Tick	Sample tick Δ between two frames in µs ⇒ sampling frequency = 1000000 / Δ	[**Long**(64-bit Integer),**Array**(Size 1)]	1 ≤
HighPassFilterType	Type of the high-pass filter (empty string if not available)	[**String**,**Scalar**]	1 ≤
HighPassFilterCutOffFrequency	Cut-off frequency of the high-pass filter (‘-1’-String if not available)	[**String**,**Scalar**]	1 ≤
HighPassFilterOrder	Order of the high-pass filter (-1 if not available)	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
LowPassFilterType	Type of the low-pass filter (empty string if not available)	[**String**,**Scalar**]	1 ≤
LowPassFilterCutOffFrequency	Cut-off frequency of the low-pass filter (‘-1’-String if not available)	[**String**,**Scalar**]	1 ≤
LowPassFilterOrder	Order of the low-pass filter (-1 if not available)	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
SensorSpacing	Distance between adjacent sensors in µm	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤

Subfolder "FrameDataEntity_x"

Navigation: /Data/Recording_x/FrameStream/Stream_x/FrameDataEntity_x

Contains all datasets of the Frame-Entity x

Datasets:

- Matrix ConversionFactors → n × m matrix of conversion factors for the sensor array
- 3-dimensional Data-Cube FrameData → cube of the frame data organized as one frame to one sample time point (n × m matrix of sampled signal values per sensor) × sample time points
  - reconstruct the value of the measured signal: y = (FrameData[x,y,t] - ADZero) * ConversionFactors[x,y]
  - reconstruct the sample time point:
- Matrix FrameDataTimeStamps → k × 3 matrix of segments where the rows are one segment and the columns are:
  - first column → time stamp of the first sample point of the segment
  - second column → first index (z-axis) of the segment in **FrameData**
  - third column → last index (z-axis) of the segment in **FrameData**

Datasets:

- none

Folder "EventStream"

Navigation: /Data/Recording_x/EventStream

(Organisational) folder for all event-based streams of this recording

Attributes:

- none

Datasets:

- none

Subfolder "Stream_x" of "EventStream"

Navigation: /Data/Recording_x/EventStream/Stream_x

Folder that contains all Event-Entities of one Event-Stream:

Attributes:

Name	Description	Data Type	StreamInfoVersion
StreamInfoVersion	Version number of the meta information structure	[**Int**(32-bit Integer),**Scalar**]	1 ≤
Label	Label	[**String**,**Scalar**]	1 ≤
SourceStreamGUID	GUID of the source stream	[**String**,**Scalar**]	1 ≤
StreamGUID	GUID of the current stream	[**String**,**Scalar**]	1 ≤
StreamType	Type of the stream **Event**	[**String**,**Scalar**]	1 ≤
DataSubType	Sub-type of the event stream (e.g. **StgSideband**, **UserInput**, **DigitalPort**)	[**String**,**Scalar**]	1 ≤

Sub-type Description:

- StgSideband → The event is associated to a STG sideband change.
- UserInput → The event is associated with an user input.
- DigitalPort → The event is associated with a digital port change.

Datasets:

- Matrix InfoEvent → n × 7 matrix of describing information vectors for the n Event-Entities:
  - Attributes: InfoVersion → Version number of the Info-Objects [**Int**(32-bit Integer),**Scalar**]

Name	Description	Data Type	InfoVersion
EventID	ID of the event entity	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
GroupID	ID of the group that the entity belongs to	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
Label	Label of the entity	[**String**,**Array**]	1 ≤
RawDataType	Type of the raw data	[**String**,**Array**]	1 ≤
RawDataBytes	Number of bytes of the raw data type	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
SourceChannelIDs	Comma separated list of ID's of (source) channel that were involved in the generation of this event	[**String**,**Array**]	1 ≤
SourceChannelLabels	Comma separated list of labels of the source channels	[**String**,**Scalar**]	1 ≤

- 2-dimensional matrix EventEntity_x → 2 × n matrix ⇒ n events with describing vector (time stamp of event, duration of event)
  - Attributes: Short description of content
  - $t_{\text{event i}} = \text{EventEntity\_}x[0, i]$ in μs
  - $\Delta_{\text{event i}} = \text{EventEntity\_}x[1, i]$ in μs

Folder "SegmentStream"

Navigation: /Data/Recording_x/SegmentStream

(Organisational) folder for all segment-based streams of this recording. A segment is a cutout of parts of the sampled signal relative to an event, defined by a pre- and post interval.

Attributes:

- none

Datasets:

- none

Subfolder "Stream_x" of "SegmentStream"

Navigation: /Data/Recording_x/SegmentStream/Stream_x

Folder that contains all Segment-Entities of one Segment-Stream:

Attributes:

Name	Description	Data Type	StreamInfoVersion
StreamInfoVersion	Version number of the meta information structure	[**Int**(32-bit Integer),**Scalar**]	$1 \leq$
Label	Label	[**String**,**Scalar**]	$1 \leq$
SourceStreamGUID	GUID of the source stream	[**String**,**Scalar**]	$1 \leq$
StreamGUID	GUID of the current stream	[**String**,**Scalar**]	$1 \leq$
StreamType	Type of the stream **Segment**	[**String**,**Scalar**]	$1 \leq$
DataSubType	Sub-type of the segment stream (e.g. **Spike**)	[**String**,**Scalar**]	$1 \leq$

Datasets:

- Matrix InfoSegment $\rightarrow$ n $\times$ 7 matrix of describing information vectors for the n Segment-Entities:
  - Attributes: InfoVersion $\rightarrow$ Version number of the Info-Objects [**Int**(32-bit Integer),**Scalar**]

Name	Description	Data Type	InfoVersion
SegmentID	ID of the segment entity	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
GroupID	ID of the group that the segment entity belongs to	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
Label	Label of the entity	[**String**,**Array**]	1 ≤
PreInterval	Time interval in µs before the segment defining event occured - definition of the beginning of the segment	[**Int**(64-bit Integer),**Array**(Size 1)]	1 ≤
PostInterval	Time interval in µs after the segment defining event occured - definition of the end of the segment length of the segment = PreInterval + PostInterval in µs	[**Int**(64-bit Integer),**Array**(Size 1)]	1 ≤
SegmentType	Type of the segment (e.g. **SpikeCutout**)	[**String**,**Array**]	1 ≤
SourceChannelIDs	Comma separated list of ID's of (source) channels that the segements are taken from → Link to the SourceChannelInfo matrix	[**String**,**Array**]	1 ≤

- 2-dimensional matrix SourceChannelInfo → n × 15 matrix ⇒ n of describing vectors for the n source channels, the structure is the same as in ChannelInfo used in section *Sub-folder "Stream_x" of "AnalogStream"*
  - Attributes: InfoVersion → Version number of the Info-Objects [**Int**(32-bit Integer),**Scalar**]
- Vector SegmentData_ts_x → n time stamps in µs of the event triggering the segment, one for each of the n segments contained by segment entity x
- 2-dimensional matrix or 3-dimensional cube SegmentData_x → k × n matrix (k sample points for one segment, n number of sampled segments) or k × m × n cube (k sample points for one segment, m number of segments for one time point/for one multi-segment, n number of sampled multi-segments) of segment data:
  - Attributes: SourceChannelID → Comma separated list of ID's of (source) channels that the segements are taken from [**String**,**Scalar**] (the same as in InfoSegment, repeated for clarification)
  - reconstruct the value of the measured segment signal (only one segment id_{segment} → 2-dimensional matrix

M[row,col]):

* $t_{ind}[row, col]$ = SegmentData ts x[col] + (row − 1) ∗ $\text{tick}_{\text{source-channel}}$ − PreInterval in µs
* $y(id_{\text{segment}}, t_{ind}(row, col))$ = (SegmentData x$[row, col]$ − $\text{ADZero}_{\text{source-channel}}$) ∗ $\text{ConversionFactor}_{\text{source-channel}} * 10^{\text{Exponent}_{\text{source-channel}}}$ in InfoChannel[source-channel].Unit

– reconstruct the value of the measured segment signal (m segments → multi-segments → 3-dimensional cube M[row,col,z]):

* col → id_{segment} → source-channel
* $t_{ind}[row, col, z]$ = SegmentData ts x[z] + (row − 1) ∗ $\text{tick}_{\text{source-channel}[col]}$ in µs
* $y(id_{\text{segment}}, t_{ind}(row, z))$ = (SegmentData x$[row, \text{col}, z]$ − $\text{ADZero}_{\text{source-channel[col]}}$) ∗ $\text{ConversionFactor}_{\text{source-channel[col]}} * 10^{\text{Exponent}_{\text{source-channel[col]}}}$ in InfoChannel[source-channel[col]].Unit

DataSubType-Average: Subfolder "Stream_x" of "SegmentStream"

Navigation: /Data/Recording_x/SegmentStream/Stream_x

Folder that contains all Segment-Entities of one Segment-Stream with **DataSybType == Average**:

Attributes: no difference to the standard case above

Datasets:

- Matrix InfoSegment: no difference to the standard case above
- Matrix SourceChannelInfo: no difference to the standard case above
- (3 × n) matrix AverageData_Range_x → (**start**, **end**, **count**) per segment average × count of segment averages contained by segment entity x. **start** and **end** denote the start and end timestamp in µs of the interval that contains all averaged segments. **count** is the number of averaged segments.
  – Attributes: description of the content
- (2 × k × n) cube AverageData_x → (mean and standard deviation) × k sample points of the segment × n number of segment averages
  – Attributes:: description of the content
  – reconstruct the value of the mean and standard deviation of the average segment (n average segments → 3-dimensional cube M[row,col,z]):
  * row: mean → row = 0; StdDev → row = 1
  * col: $t_{ind}(col)$ = (col − 1) ∗ $\text{tick}_{\text{source-channel}}$ → time range $(0, PreInterval[\text{SegmentID}] + PreInterval[\text{SegmentID}])$ in µs
  * z: z = id_{average} (number of average segment)
  * $Mean(id_{\text{average}}, t_{ind}(col))$ = (AverageData x$[0, \text{col}, id_{\text{average}}]$ − $\text{ADZero}_{\text{source-channel}}$) ∗ $\text{ConversionFactor}_{\text{source-channel}} * 10^{\text{Exponent}_{\text{source-channel}}}$ in $\text{InfoChannel}_{\text{source-channel}}$.Unit
  * $StdDev(id_{\text{average}}, t_{ind}(col))$ = AverageData x$[1, \text{col}, id_{\text{average}}]$ ∗ $\text{ConversionFactor}_{\text{source-channel}}$ ∗ $10^{\text{Exponent}_{\text{source-channel}}}$ in $\text{InfoChannel}_{\text{source-channel}}$.Unit

Folder "TimeStampStream"

Navigation: /Data/Recording_x/TimeStampStream

(Organisational) folder for all TimeStamp-based streams of this recording

Attributes:

- none

Datasets:

- none

Subfolder "Stream_x" of "TimeStampStream"

Navigation: /Data/Recording_x/TimeStampStream/Stream_x

Folder that contains all TimeStamp-Entities of one TimeStamp-Stream:

Attributes:

Name	Description	Data Type	StreamInfoVersion
StreamInfoVersion	Version number of the meta information structure	[**Int**(32-bit Integer),**Scalar**]	$1 \leq$
Label	Label	[**String**,**Scalar**]	$1 \leq$
SourceStreamGUID	GUID of the source stream	[**String**,**Scalar**]	$1 \leq$
StreamGUID	GUID of the current stream	[**String**,**Scalar**]	$1 \leq$
StreamType	Type of the stream **TimeStamp**	[**String**,**Scalar**]	$1 \leq$
DataSubType	Sub-type of the TimeStamp stream (e.g. **NeuralSpike**)	[**String**,**Scalar**]	$1 \leq$

Sub-type Description:

- **NeuralSpike** → The entity contains time stamps of neural spikes

Datasets:

- Matrix InfoTimeStamp → n × 7 matrix of describing information vectors for the n Event-Entities:
  - Attributes: InfoVersion → Version number of the Info-Objects [**Int**(32-bit Integer),**Scalar**]

Name	Description	Data Type	InfoVersion
TimeStampEntityID	ID of the event entity	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
GroupID	ID of the group that the entity belongs to	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
Label	Label of the entity	[**String**,**Array**]	1 ≤
Unit	Physical unit of the measured sensor value	[**String**,**Array**]	1 ≤
Exponent	Exponent n ⇒ 1En resp. 10^n in which the channel values magnitude is measured (e.g. k,m,µ,...)	[**Int**(32-bit Integer),**Array**(Size 1)]	1 ≤
SourceChannelIDs	Comma separated list of ID's of (source) channel that were involved in the generation of this event	[**String**,**Array**]	1 ≤
SourceChannelLabels	Comma separated list of labels of the source channels	[**String**,**Scalar**]	1 ≤

- Vector TimeStampEntity_x → n time stamps in μs

Comment

All time-related information except dates (100ns ticks) are given in μs ticks!!

Category:Software

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX