3 Brain

Documentation for BRW v. 3.x and BXR v. 2.x File Formats
(available in BrainWave v 3.x and v4.x)

Document version 1.2.0

February 2021

Contents

1. Introduction 5

2. Overview 6

2.1. Formatting conventions 6

2.2. HDF Objects 6

2.3. Channel notation 7

2.4. Compound data types 7

2.4.1. Channel 7

2.4.2. Channels Group 7

2.4.3. Version 7

2.4.4. Spike Detection Parameters 8

2.4.5. Spike Waveform Parameters 9

2.4.6. Spike Sorting Parameters 9

2.4.7. Lfp Detection Parameters 10

2.5. General information 11

3. BRW-File 12

3.1. Root 12

3.2. 3BData 13

3.2.1. Event Based Raw Ranges (Spikes Based Raw Ranges until v.102) 14

3.2.2. Array Wide Raw Ranges 15

3.2.3. Wavelet Coefficients 15

3.3. 3BInfo 16

3.4. 3BNoise 16

4. BXR-File 17

4.1. Root 17

4.2. 3BResults 18

4.3. 3BArrayEvents 19

4.4. 3BChEvents 20

4.5. 3BInfo 23

4.6. 3BArtifacts 24

4.7. 3BNoise 25

4.8. 3BSpikes 26

4.9. 3BLFPs 27

4.10. 3BEFPStimuli 28

5. Shared Information 29

5.1. 3BUserInfo 29

5.2. 3BRecInfo 30

5.3. 3BMeaChip 31

5.4. 3BMeaStreams 32

5.5. 3BMeaSystem 33

5.6. 3BRecVars 34

5.7. 3BSourceInfo 36

1. Introduction

This document provides information on the 3Brain’s data file formats based on Hierarchical Data Format 5 (HDF5), namely BRW-File (3Brain Raw data file for high resolution MEA platform) up to version 3.2, and BXR-File (3Brain Experiment Results file for high resolution MEA platform) up to version 2.1. These BRW-Files are available for writing in BrainWave v. 3.x and BrainWave v. 4.x. They are pure HDF5 files with a .brw or .bxr extension and with a custom data organization and metadata information in order to allow BrainWave read/write operations.

HDF5 is a file format designed to store large and complex data collections that are organized in a filesystem-like structure and that can be accessed with a POSIX-like syntax */path/to/resource*. HDF5 has no limit on the number and size of data objects, is completely portable, can be used even on massive parallel systems and comes with an API for C, C++, Fortran 90, Java and CLI .NET. More information on HDF5, its data structure, tools and software and example programs to read and write to it can be found on http://www.hdfgroup.org/.

This document describes data organization inside HDF5-based BRW- and BXR-Files.

2. Overview

2.1. Formatting conventions

The following formatting conventions are used in this document:

- Important information that can compromise the integrity and hence the possibility to properly read the files are in red and underlined
- Optional structure objects are marked with an [optional]
- Main used compound data types are in blue and underlined
- A paragraph corresponding to a 3bgroup (see next section) has a section for each available version, such as Version 200
- In file-tree graphic representation, Groups are indicated with gray-filled rounded-corner rectangles (group), Datasets with white and bordered rounded-corner rectangles (dataset) and Attributes with dashed-bordered rectangles (attribute). Finally, in case the object in the file-tree can vary according to a given condition, a orange-filled rounded-corner rectangle indicates such conditional object (conditional)

2.2. HDF Objects

The following HDF objects are used to define the structures of BRW- and BXR-files:

- Group: a collection of objects, i.e. dataset and other groups
- Dataset: a 1- or multi-dimension array of data elements storing numerical data
- Attribute: metadata information associated to a group or a dataset

Groups, which can be seen as the folders in a filesystem, can be either simple groups containing non-versionable objects or versionable groups containing objects whose names, layout and organization depends from the group's version. Versionable groups have at least one Attribute, named Version, which defines the version for their content and might have additional attributes defining optional layout information for their content. All versionable groups are prefixed with 3B (standing for 3Brain) and will be referred here on as 3bgroups. The file root group ('/') is also a versionable group having a Version Attribute defining the general version of the file.

In addition to read the information contained in BRW- and BXR-Files, some users might be interested also in write to them by adding their specific data. In this case it is recommended to add data in separate groups (folders) from the 3bgroup ones in order to avoid any conflict with the 3Brain's data structure which might compromise the correct loading of the files in BrainWave.

2.3. Channel notation

A MEA chip channel (electrode) can be represented either with a linear index, i.e. an ID, or by means of its row and column subscripts. Supposing to have 4096 channels in a 64 x 64 grid, the linear index ranges in [0 4095], being 0 the first channel and 4095 the last one. When using instead the subscript notation, the channels are identified by their row and column, which both vary in the range [1 64]. Hence by subscript the first channel is the channel (1, 1), while the last one is the channel (64, 64).

2.4. Compound data types

Here are listed main compound data types that are used in different locations inside the BRW- and BXR-Files.

2.4.1. Channel

A channel can be represented either with a linear index, i.e. an ID, or by means of its row and column subscripts. When using the subscript notation, the BwCh is represented as a compound data type made up by two fields:

- Row: the row subscript of the channel comprised between 1 and the number of rows of the MEA
- Col: the column subscript of the channel comprised between 1 and the number of columns of the MEA

2.4.2. Channels Group

A group (collection) of channels is represented with a compound data type, BwChsGroup, having the following fields:

- Name: a scalar string for the list name
- Color: a compound data type for the list color made up by:
  - KnownColor: an integer value whether the color corresponds to one of the .NET System.Drawing.KnownColor; otherwise, 0
  - Alpha: the alpha component of the color
  - Red: the red component of the color
  - Green: the green component of the color
  - Blue: the blue component of the color
- Chs: a one-dimensional array of BwCh representing the channels belonging to the Channels Group
- IsVisible [optional]: a Boolean indicating whether the channel list is visible or not

Since versions 3.0.1(brw) and 2.0.1(bxr), BwChsGroup has an additional field:

- Units: a one-dimensional array of integer representing the units belonging to the Channels Group

2.4.3. Version

A Version compound data type has the following fields:

- Major: the major component of the version number
- Minor: the minor component of the version number
- Build: the build component of the version number or -1 whenever not used
- Revision: the revision component of the version number or -1 whenever not used

2.4.4. Spike Detection Parameters

Spike detection parameters are stored in a BwSpikeDetParams compound data type with the following fields:

- Algorithm: an integer value indicating the used spike detection algorithm according to the following enumeration:

```
enum SpikeDetectionAlgorithm
{
     [Description("PTSD (Precise Timing Spike Detection)")]
     Ptsd = 0,
     [Description("SWDT (Sliding Window Differential Threshold)")]
     Swdt = 1,
     [Description("HT (Hard Threshold)")]
     HardThr = 2,
     [Description("QBD (Quantile Based Detection)")]
     Qbd = 3
}
```

- NoDetectionScopeOnArtifactRegions: determines the scope where spikes are not detected for time intervals comprised inside the artifact regions
- AssignmentRule: an integer value indicating the rule used to assign the spike time instant according to the following enumeration:

```
enum SpikeAssignmentRule : byte
{
      [Description ("Negative Peak")]
      NegPeak = 0,
      [Description ("Positive Peak")]
      PosPeak = 1,
      [Description ("Higher Peak")]
      HigherPeak = 2
}
```

- STDFactor: a floating-point value indicating the standard deviation multiplication factor used to detect spikes
- SWDTSlidingWindowLength: a floating-point value indicating the length in ms of the windows used to detect spikes (relevant only for SWDT algorithm)
- PTSDPeakLifetimePeriod: a floating-point value representing the lifetime period in ms (relevant only for PTSD algorithm)
- PTSDOvershoot: a floating-point value for the overshoot in ms (relevant only for PTSD algorithm)
- RefractoryPeriod: a floating-point value indicating the refractory period in ms
- QBDThreshold: a floating-point value for the threshold used to detect spikes (relevant only for QBD algorithm)
- QBDMinAvgAmp: a floating-point value for the minimum average amplitude (relevant only for QBD algorithm)
- QBDMinLenght: a floating-point value indicating the minimum length of the depolarization for detecting a spike (relevant only for QBD algorithm)
- QBDMaxLenght: a floating-point value indicating the maximum length of the depolarization for detecting a spike (relevant only for QBD algorithm)
- QBDRepolarizationThreshold: a floating-point value indicating the threshold to identify the repolarization (relevant only for QBD algorithm)

2.4.5. Spike Waveform Parameters

Spike waveform parameters are stored in a BwSpikeWaveformParams compound data type with the following fields:

- PrePeriodMs: a floating-point value indicating the period in ms that is recorded before the spike event
- PostPeriodMs: a floating-point value indicating the period in ms that is recorded after the spike event
- DiscardIncomplete: a Boolean value indicating if incomplete waveforms are discarded or not

2.4.6. Spike Sorting Parameters

Spike sorting parameters are stored in a BwSpikeSortingParams compound data type with the following fields:

- PreEventMs: a floating-point value indicating the period in ms before the spike event that is considered by the sorting algorithm
- PostEventMs: a floating-point value indicating the period in ms after the spike event that is considered by the sorting algorithm
- FeatureExtractionMethod: an integer value indicating the used feature extraction algorithm according to the following enumerator:
```
enum SpikeSortingFeatureExtractionMethods
{
      [Description("PCA")]
      PCA,
      [Description("Spike Characteristics")]
      SpikeCharacteristics
}
```
- FeatureSelection: an integer value indicating the used feature selection algorithm according to the following enumerator:
```
enum SpikeSortingFeaturesSelection
{
      Feat_1_2_3,
      Feat_1_2,
      Feat_1_3,
      Feat_2_3
}
```
- ClusteringMethod: an integer value indicating the used feature selection algorithm according to the following enumerator:
```
enum ClusteringMethod
{
      [Description("k-means & Silhouette")]
      KMeans_Silhouette,
      [Description("k-means & PBM-index")]
      KMeans_PBM,
      [Description("PCA")]
      MixturesOfGaussiansClassifier
 }
```

- ClusteringMethodMinNSpikesPerCluster: an integer value indicating the minimum number of spikes that a cluster must contain to be valid
- ClusteringMethodMinNClusters: an integer value indicating the minimum number of clusters that must be defined by the sorting algorithm

- ClusteringMethodMaxNClusters: an integer value indicating the maximum number of clusters that can be defined by the sorting algorithm
- ClusteringMethodDiscardOutliers: a Boolean value indicating if the sorting algorithm is discarding outliers
- ClusteringMethodOutliersThresholdNSd: a floating-point value indicating the standard deviation of the distance from the centroid used to find outliers
- RemoveDuplicates: a Boolean value indicating it the removal of duplicates has been applied
- DuplicatesScope: an integer value indicating the scope in which duplicate removal has been applied, according to the following enumerator:

```
enum ChNeighborsScope
{
      [Description("no scope")]
      Nothing,
      [Description("no neighbors")]
      NoNeighbors,
      [Description("4 channels (up,down,left,right")]
      Nearest4,
      [Description("8 channels (grid 3x3")]
      Nearest8,
      [Description("24 channels (grid 5x5)")]
      Nearest24,
      [Description("48 channels (grid 7x7)")]
      Nearest48,
      [Description("80 channels (grid 9x9)")]
      Nearest80,
      [Description("120 channels (grid 11x11)")]
      Nearest120,
      [Description("168 channels (grid 13x13)")]
      Nearest168,
      [Description("All channels")]
      All
}
```

- CrossCorrWindow: a floating-point value indicating the duration of the window of cross correlation, in ms
- CrossCorrBin: a floating-point value indicating the duration of the bin of cross correlation, in ms
- CrossCorrThreshold: a floating-point value indicating the threshold applied to the cross correlation

2.4.7. Lfp Detection Parameters

Lfp detection parameters are stored in a BwLfpDetParams compound data type with the following fields:

- Algorithm: an integer value indicating the used spike detection algorithm according to the following enumeration:

```
 enum LfpDetectionAlgorithm
 {
      [Description("Hard Threshold")]
      HardThreshold = 0
 }
```

- HighThreshold: a double-precision floating-point value for the high threshold used to detect lfps in µV
- LowThreshold: a double-precision floating-point value for the low threshold used to detect lfps in µV

- WindowLength: a floating-point value indicating the duration of the sliding window used to detect lfps in ms
- RefractoryPeriod: a floating-point value indicating the refractory period in ms

2.5. General information

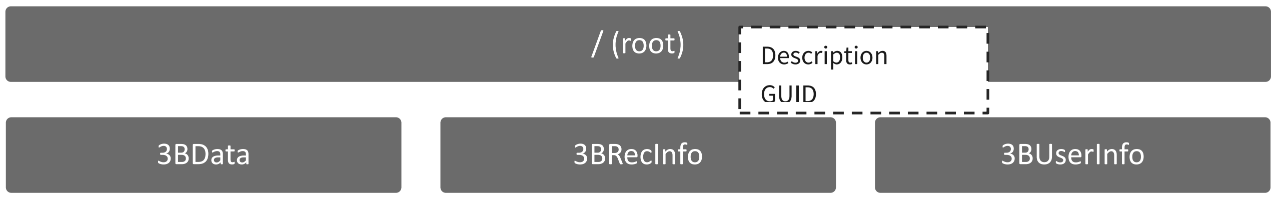
In what follows, each 3BGroup will be listed by providing its minimum version, the current version and details on its content for each different available version. Version attribute value must always be comprised between the minimum and the current version in order to avoid file opening failures. Only additional attributes apart from the version attribute will be listed in the following paragraphs.

3. BRW-File

3.1. Root

Minimum Version: 300
Current Version: 320

Version 320



Additional Attributes

- Description: a string description of the file containing general information such as the file type and the creation date. The description must start with a “BRW-File Level3” prefix, to avoid opening failures
- GUID: a global unique identifier that uniquely identifies the file

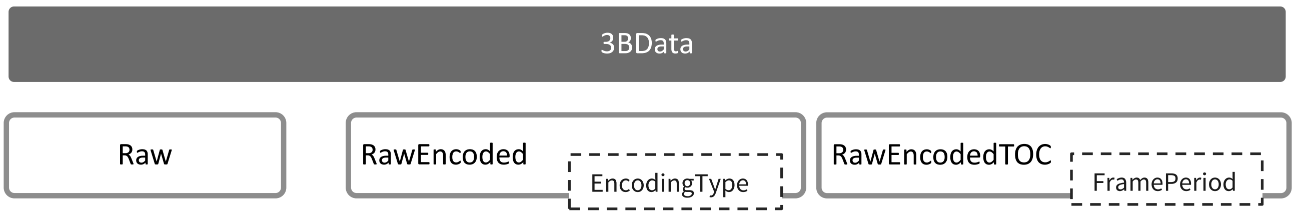
Contained objects

- 3BData: contains recorded data
- 3BRecInfo: contains information on the recording such as MEA chip properties and recording parameters. Refer to paragraph 5.2 for information on its content
- 3BUserInfo: contains user information. Please refer to paragraph 0

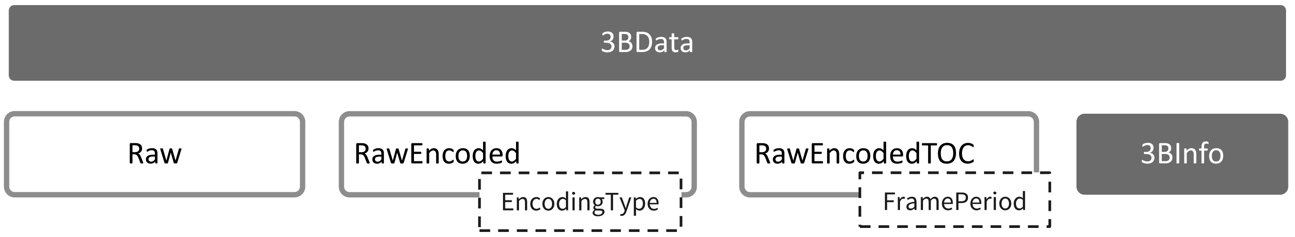
3.2. 3BData

Minimum Version: 100
Current Version: 102

Version 100 and 101



Version 102



Contained Objects

- Raw [optional].
  - Version 100: a T x W matrix of integer values, where T is the number of recorded samples and W the number of recorded channels from the device stream (stream of data coming from the microelectrode array), representing the electrophysiological recorded data for each recording electrode as digital values. Conversion from recorded sample index to time information and from digital values to analog values can be performed by using the recording variables (see paragraph 5.6).
  - Version 101: a N x 1 matrix of integer values, where $N = T \cdot W$, is the number of recorded samples and W the number of recorded channels. For each acquired sample from the device stream, channels are placed in the matrix one after the other (C_1, C_2, …C_T in sample S_1; C_1, C_2, …C_T in sample S_2; and so on up to sample S_W).
- RawEncoded [optional]: a 1-dimensional array of values encoding the recorded samples coming from the device stream according to the EncodingType Attribute (see below for further details).
  - Version 102: If this dataset is present and EncodingType is EventsBasedRawRanges or ArrayWideRawRanges then the RawEncodedTOC dataset must be present too.
- RawEncodedTOC [optional]: a Table Of Content (TOC) that together to the FramePeriod Attribute helps on navigating the RawEncoded dataset. The TOC lists the positions in number of bytes inside the RawEncoded dataset at which a frame in the device stream, as a multiple of the FramePeriod value, is present. For instance, having a TOC with N values and a FramePeriod of F frames, the i-value inside the TOC (i defined in range [0 N-1]) gives the position in number or bytes of the frame $i * F$ inside the RawEncoded array.
- 3BInfo (from Version 102): contains the noise informations.

Details on RawEncoded

- Version 101: There is only one type of encoding supported and consequently the EncodingType Attribute can only have the value SpikesBasedRawRanges.
- Version 102: There are three types of encoding supported:

- EventsBasedRawRanges
- ArrayWideRawRanges
- WaveCoeffs

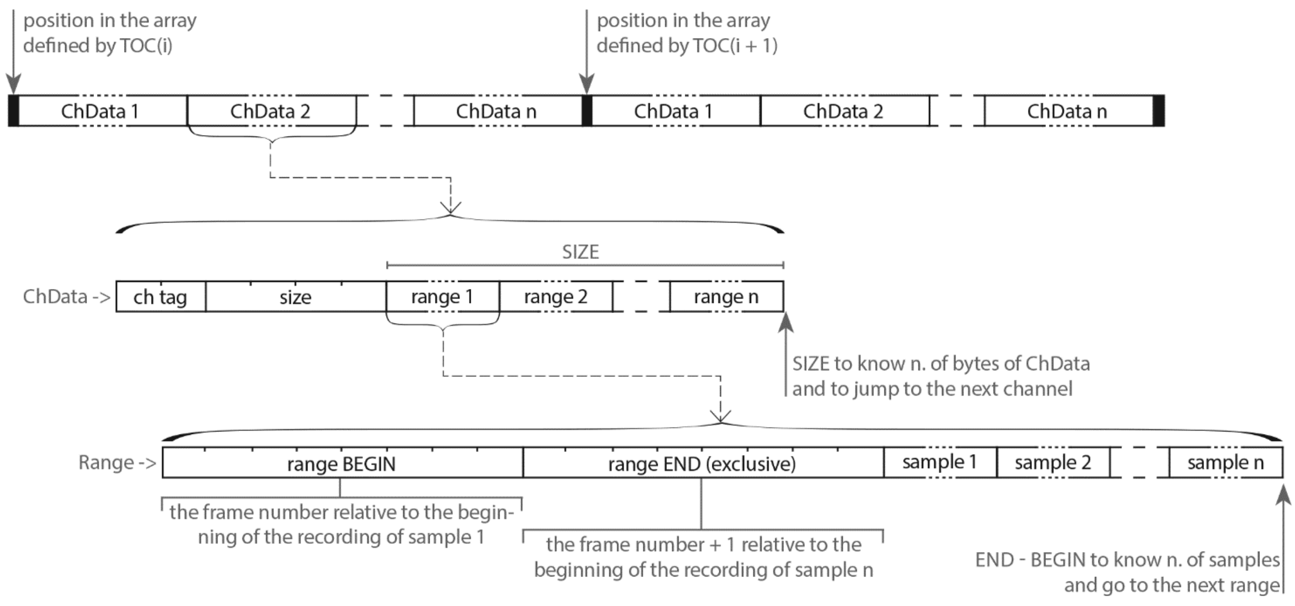
See following paragraphs for details.

3.2.1. Event Based Raw Ranges (Spikes Based Raw Ranges until v.102)

This type of encoding defines the ranges of data (raw ranges) that are saved from the stream according to the identified events. The saved ranges of data depend from the algorithm parameters and typically consist in samples around the actual timestamp of the identified spikes. The remaining portions of the stream, i.e. those were no spikes are found, are instead discarded.

The RawEncoded array consists of a series of bytes that define the raw ranges saved for each channel. In particular, referring to the below picture, at each position in the array defined in the TOC, a series of ChData elements will be found for each channel whose data has been stored.

A ChData consists of a small header of 6 bytes, where the first 2 bytes encode with an `UInt16` the channel tag, i.e. the linear index of the channel in the MEA array, and the following 4 bytes (`Int32`) represent the size of the ChData (without including the header itself). Size can be used to jump to the following ChData. The following bytes in the ChData element encodes the raw ranges saved for the channel as Range elements.
Each Range then consists of a small header with 2 `Int64` values (8 bytes each) that indicate the **begin** of the range in number of frames (relative to the beginning of the recording) and the **end** (exclusive) of the range in number of frames . The difference between **end** and **begin** gives the number of sample that will following in the Range element. Each sample represents the electrophysiological recorded data as digital values. Conversion from recorded sample index to time information and from digital values to analog values can be performed by using the recording variables (see paragraph 5.6).

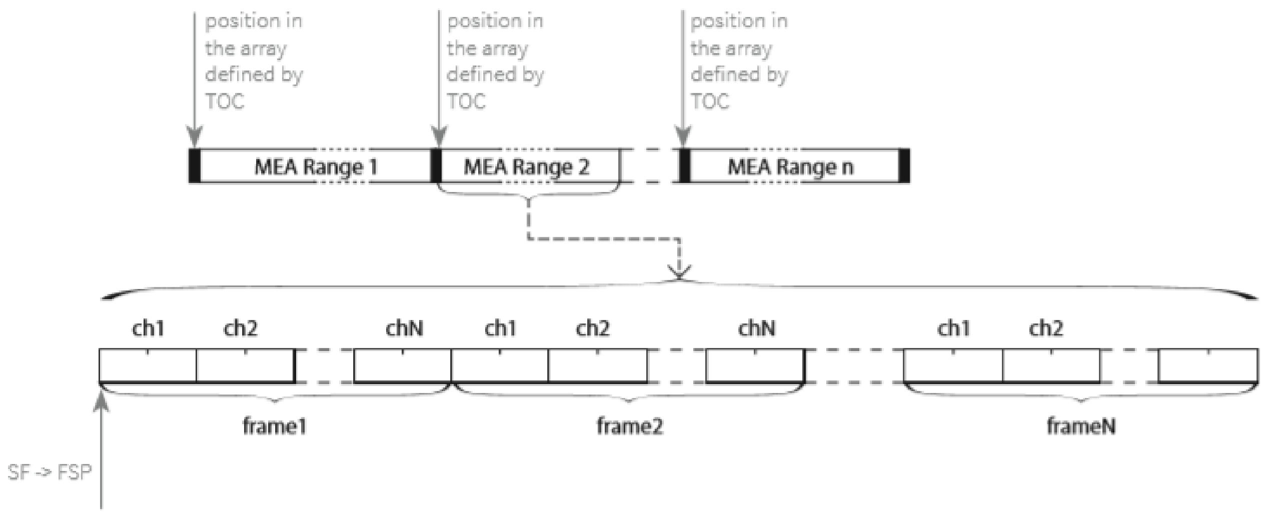

position in the array defined by TOC(i)
position in the array defined by TOC(i + 1)
ChData 1
ChData 2
ChData n
ChData 1
ChData 2
ChData n
SIZE
ChData ->
ch tag
size
range 1
range 2
range n
SIZE to know n. of bytes of ChData and to jump to the next channel
Range ->
range BEGIN
range END (exclusive)
sample 1
sample 2
sample n
the frame number relative to the beginning of the recording of sample 1
the frame number + 1 relative to the beginning of the recording of sample n
END - BEGIN to know n. of samples and go to the next range

3.2.2. Array Wide Raw Ranges

This type of encoding defines the ranges of data (raw ranges) that are saved from the stream according to array-wide events, like stimulations, or user-defined intervals, defined in a protocol. In the first case, the saved ranges of data depend from the algorithm parameters and typically consist in samples around the timestamp of the stimulation. The remaining portions of the stream are instead discarded.

The RawEncoded array consists of a series of bytes that define the raw ranges saved for the whole array. As illustrated in the following picture, the file stream position (FSP) of the beginning of each MEA range is stored in the TOC, together with the corresponding starting frame (SF) and number of frames recorded (N).

Each range consists of `M x N` values, where `M` is the number of channels recorded, and `N` is, as stated above, the number of samples (frames) per range. Each sample represents the electrophyisiological recorded data as digital values. Conversion from recorded sample index to time information and from digital values to analog values can be performed by using the recording variables (see paragraph 5.6).


position in the array defined by TOC
position in the array defined by TOC
position in the array defined by TOC
MEA Range 1
MEA Range 2
MEA Range n
ch1
ch2
chN
ch1
ch2
chN
ch1
ch2
frame1
frame2
frameN
SF -> FSP

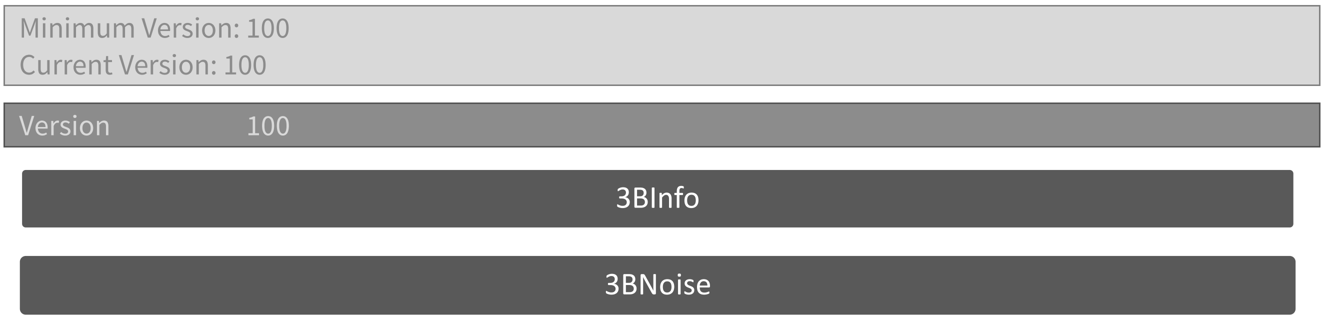
3.2.3. Wavelet Coefficients

In this type of encoding, raw data is continuously stored in ranges as an array of coefficients (wavelet coefficients) for each recorded channel.

The RawEncoded array consists in a sequence of coefficients that can be used to rebuild the raw information, saved for the whole array.

Each range consists of `M x N` values, where `M` is the number of channels recorded, and `N` is the number of wavelet coefficients. Conversion from recorded data to analog values is performed by using the recording variables and the wavelet parameters.

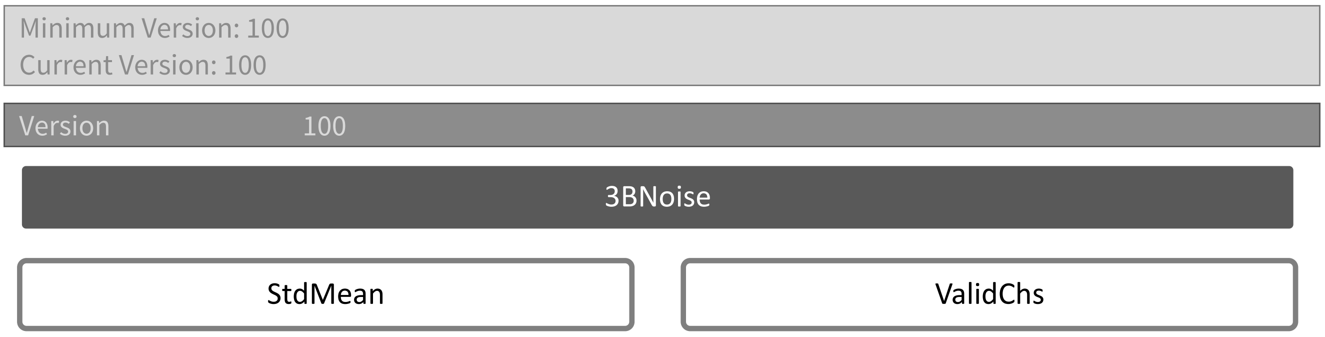
3.3. 3BInfo



Contained objects

- 3BNoise: a group with information on the channels' measured noise

3.4. 3BNoise



Contained objects

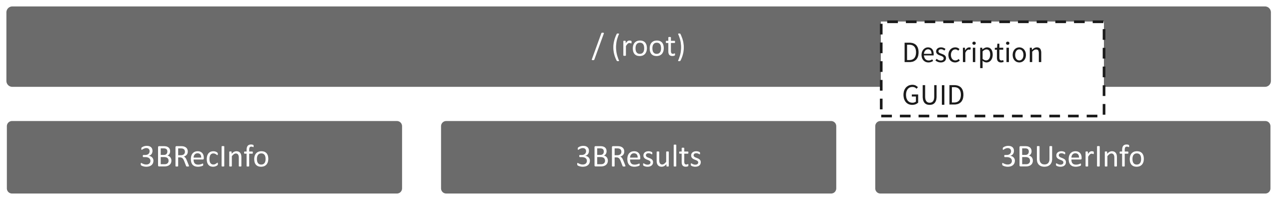
- StdMean: a compound data type containing the measured values of standard deviation and of mean for each channel. Values are referred to digital samples values and in order to convert please refer to section 5.6
- ValidChs: the list of Channels whose noise values have been considered valid. During recording, some channels showing noise values outside the normal distribution might be considered outliers

4. BXR-File

4.1. Root

Minimum Version: 200
Current Version: 211

Version	from 200 to 211



Additional Attributes

- Description: a string description of the file containing general information such as the file type and the creation date. The description must start with a "BXR-File Level2" prefix, to avoid opening failures
- GUID: a global unique identifier that uniquely identifies the file

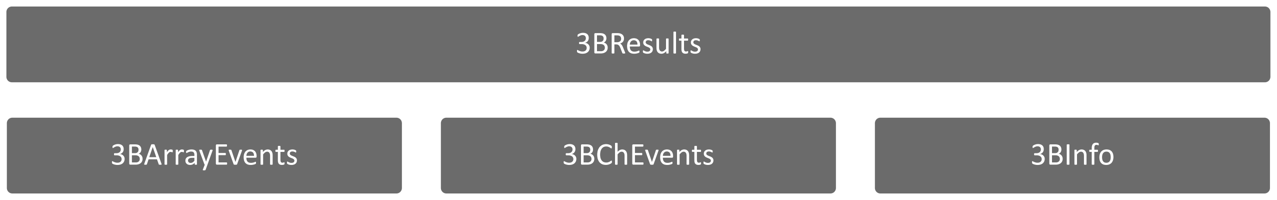
Contained objects

- 3BRecInfo: contains information on the recording such as MEA chip properties and recording parameters. Refer to paragraph 5.2 for information on its content
- 3BResults: data results coming from data analysis on raw data
- 3BUserInfo: User information added to the data. Please refer to paragraph 0

4.2. 3BResults

Minimum Version: 100
Current Version: 101

Version 100 and 101

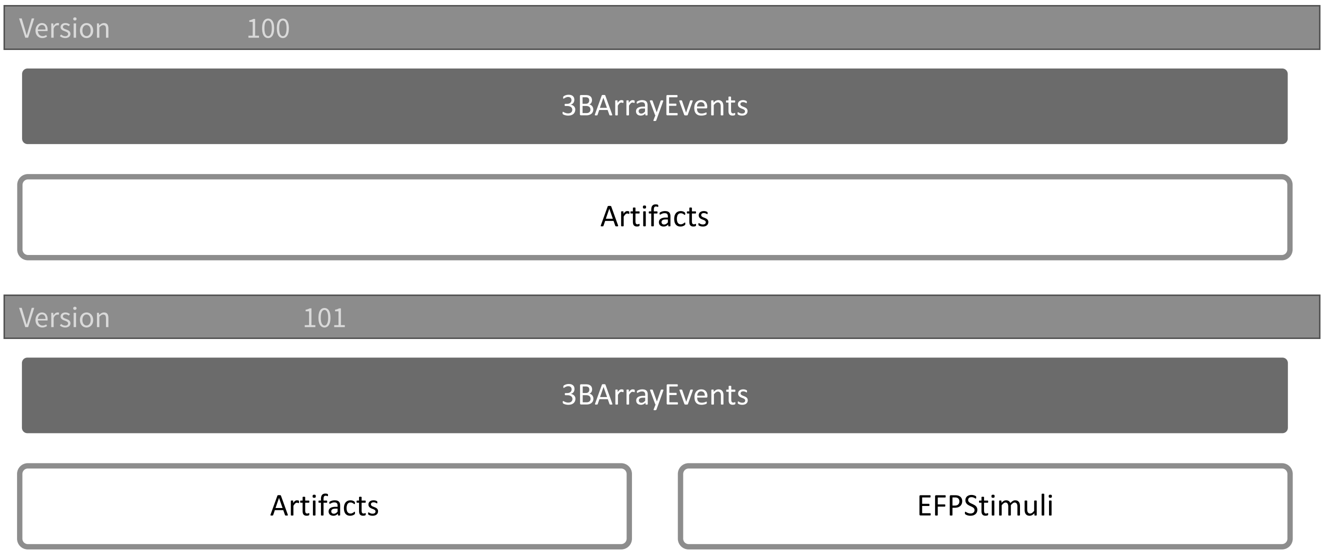


Contained objects

- 3BArrayEvents: contains results on array-wide events (e.g., EFP stimuli)
- 3BChEvents: contains results on channel-based events (e.g. spikes, bursts, lfps)
- 3BInfo: contains information on results, such as lookup tables and algorithm parameters

4.3. 3BArrayEvents

Minimum Version: 100
Current Version: 101

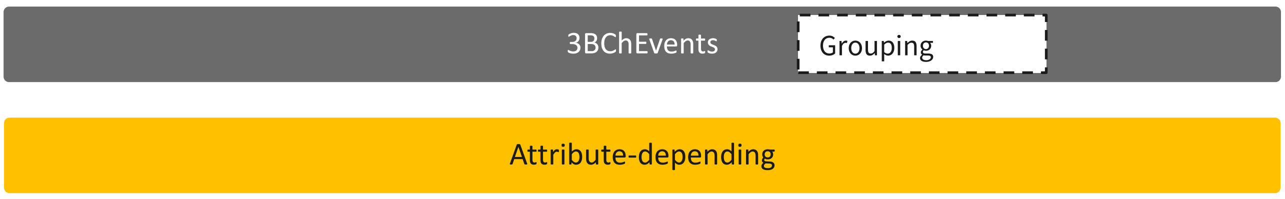


Contained objects

- Artifacts: an array of compound data type with the following fields:
  - Begin: the time instant considered as the beginning of the artifact region
  - Actual: the time instant of the artifact
  - End: the time instant considered as the end of the artifact region
  - ChID: the channel ID (linear index) identifying the reference channel on which the artifact has been identified. Artifact might be identified on multiple channels in case a complex stimulation pattern is used to deliver bipolar stimulation on different stimulating channels
- EFPStimuli: an array of compound data type with the following fields:
  - Time: the time instant of the stimulation
  - Label: the name of the protocol phase that the stimulus belongs to
  - Step: the step of the protocol phase that the stimulus belongs to
  - Color: the color associated with the protocol phase

4.4. 3BChEvents

Minimum Version: 100
Current Version: 103



Additional attributes

- Grouping: determines the structure of the contained objects. Possible values are:
  - ByChannel: channel events are grouped by channel
  - Merged: channel events are merged together and ordered according to time scale

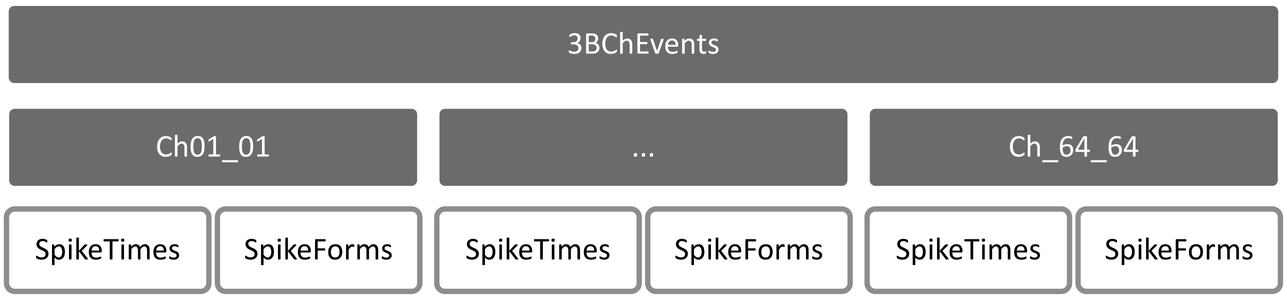
From Version 101, Grouping can only be Merged

Contained objects

Contained objects depends on the Grouping attribute

Case: Grouping = ByChannel

Version (3BChEvents) 100



Contained objects

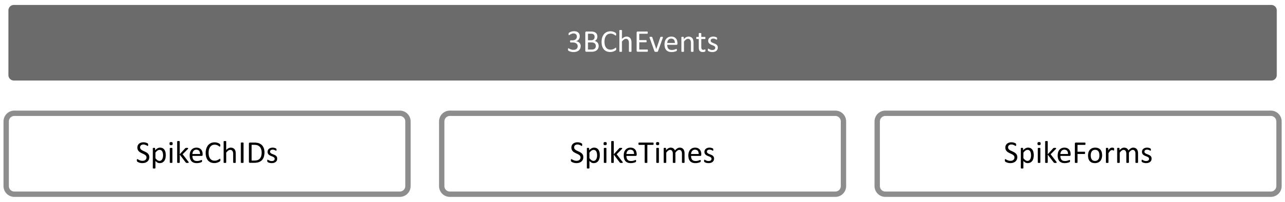
A channel-specific group is present for each spike detected channel. Each group is named in the form `ChRR_CC`, where `RR` represents the row of the channel (in the range [01 64]) and `CC` the column of the channel (in the range [1 64]).

Each channel group contains the following information related to that channel:

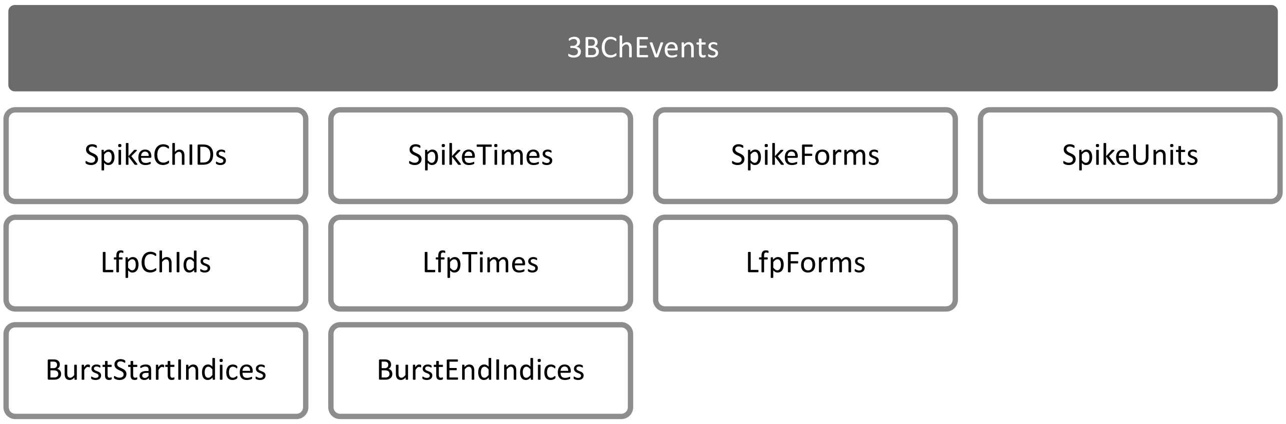
- SpikeTimes: a one-dimensional array of integer values, where values represent the time instants in number of frames of identified spikes
- SpikeForms [optional]: a two-dimensional array `N x S` defining the waveforms of the spikes and where `N` is the number of identified spikes (must equal the length of SpikeTimes) and `S` is the number of samples for each spike. Each row represents thus the waveform of the corresponding spike

Case: Grouping = Merged

Version (3BChEvents) 100 and 101



Version(3BChEvents) 102 and 103



Contained objects

Each set of datasets (SpikeXXX, LfpXXX, BurstXXX) is required only if the corresponding data analysis has been performed. Otherwise, they sould not be written. LfpXXX datasets are required when the Experiment Type is EFP.

- SpikeChIDs: a one-dimensional array of Channel ID for each entry present in SpikeTime
- SpikeTime: a one-dimensional array of integer values, where values represent the time instants in number of frames of identified spikes for all the spike detected channels. The length of SpikeTime must equal the length of SpikeChIDs
- SpikeForms [optional]:
  - Version 100: a two-dimensional array `N x S` defining the waveforms of the spikes for each entry present in SpikeTime. `N` is the number of identified spikes (must equal the length of SpikeTimes) and `S` is the number of samples for each spike. Each row represents thus the waveform of the corresponding spike
  - Version 103: a one-dimensional array of length `M`, where `M` equals `N x S` as seen above. It thus contains the waveforms for each detected spike, muxed into one dimension
- SpikeUnits [optional]: a one-dimensional array of integer values for each entry present in SpikeTime
- LfpChIDs: a one-dimensional array of Channel ID for each entry present in LfpTime
- LfpTime: a one-dimensional array of integer values, where each value represents the time instant, in frames, of each identified lfp for all lfp detected channels. The length of LfpTime must equal the length of LfpChIDs
- LfpForms:
  - Version 102: a two-dimensional array `N x S` defining the waveforms of the lfps for each entry present in LfpTime. `N` is the number of identified lfps (must equal the length of

LfpTimes) and `S` is the number of samples for each lfp. Each row represents thus the waveform of the corresponding spike

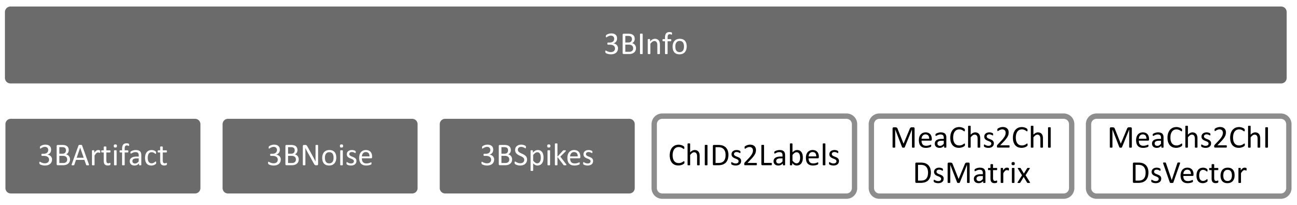
  - Version 103: a one-dimensional array of length `M`, where `M` equals `N x S` as seen above. It thus contains the waveforms for each detected lfp, muxed into one dimension

- BurstStartIndices: a one-dimensional array of integer values, containing the indexes in the SpikeXXX datasets of the first spike of each burst. The length of BurstStartIndices must equal that of BurstEndIndices
- BurstEndIndices: a one-dimensional array of integer values, containing the indexes in the SpikeXXX datasets of the last spike of each burst. The length of BurstEndIndices must equal that of BurstStartIndices

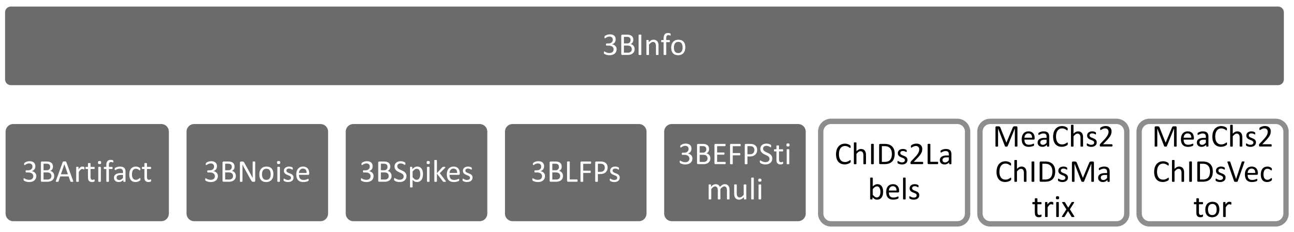
4.5. 3BInfo

Minimum Version: 100
Current Version: 101

Version 100



Version 101

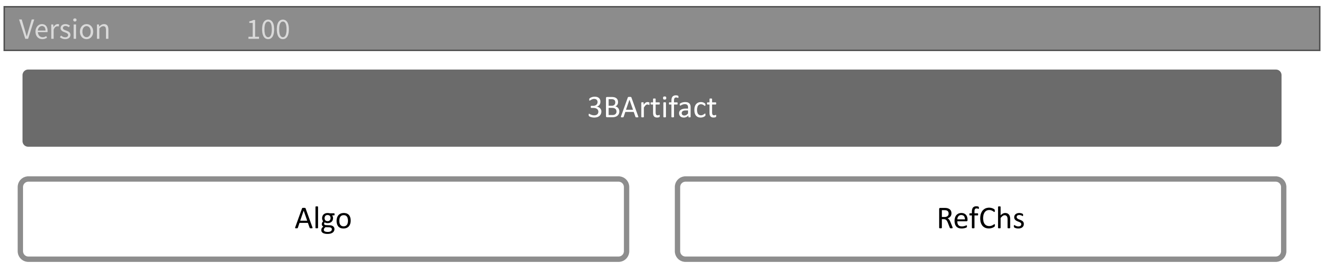


Contained objects

- 3BArtifact [optional]: contains information on eventually performed artifact detection
- 3BNoise [optional in Version 100]: a group with information on the channels' measured noise
- 3BSpikes: a group containing information on the spike results
- 3BLFPs: a group containing information on the lfp results
- 3BEFPStimuli: a group containing information on the efp stimulation
- ChIDs2Labels: a one-dimensional array that represents a lookup table to convert a channel ID to a channel label. The indexed position in the array corresponds to a channel ID and the contained value to the channel label
- MeaChs2ChIDsMatrix: a `N x M` matrix of integer values that represents a lookup table to convert from channel subscripts to channel ID, where `N` and `M` are the number of rows and of columns respectively of the MEA chip. Be careful on the indexing convention, i.e. either zero-based indexing or one-based indexing, used by your environment. The ID for the channel identified by first row and first column (subscript (1, 1), see section 2.3, and label Ch01_01), can be found at position `Matrix[0, 0]` in a zero-based indexing environment or at position `Matrix[1, 1]` in a one-based indexing environment. If this is confusing you can opt to refer to MeaChs2ChIDsMatrix in order to convert from channel subscripts to channel IDs
- MeaChs2ChIDsVector: a `W x 3` matrix that represents a lookup table to convert from channel subscripts to channel IDs, where `W` is the number of channels (electrodes) of the MEA chip and the three columns represents the channel row subscript, the channel column subscript and the channel ID respectively

4.6. 3BArtifacts

Minimum Version: 100
Current Version: 100



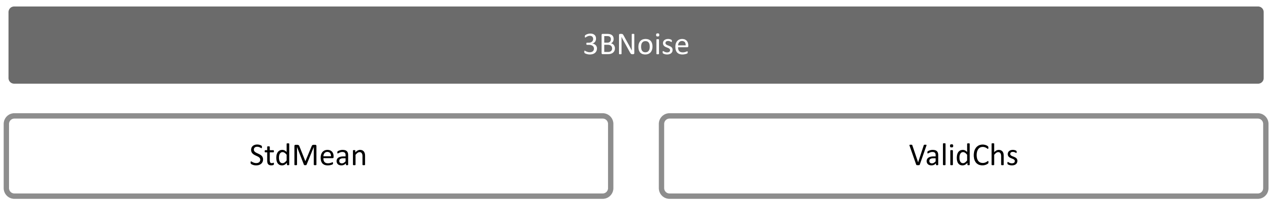
Contained objects

- Algo: a value for the algorithm used for artifact detection
- RefChs: a compound data type for each channel used as reference channel for detecting artifacts (a reference channel is a channel that has been taken in order to search for artifacts). The compound data type contains the row subscript, the column subscript and the threshold used for the reference channel. In case of automatic detection, there is only one reference channels; in case of manual detection, the user may have chosen more reference channels

4.7. 3BNoise

Minimum Version: 100
Current Version: 100

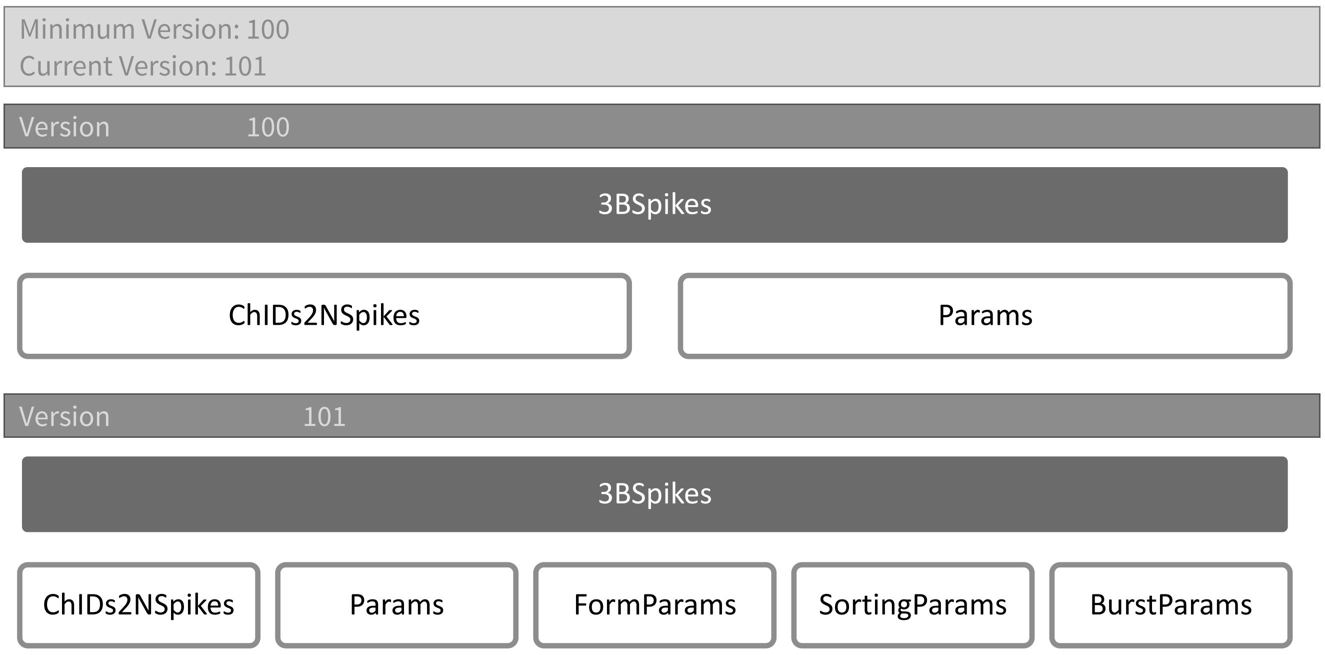
Version 100



Contained objects

- StdMean: a compound data type containing the measured values of standard deviation and of mean for each channel. Values are referred to digital samples values and in order to convert please refer to section 5.6
- ValidChs: the list of Channels whose noise values have been considered valid. During recording, some channels showing noise values outside the normal distribution might be considered outliers

4.8. 3BSpikes



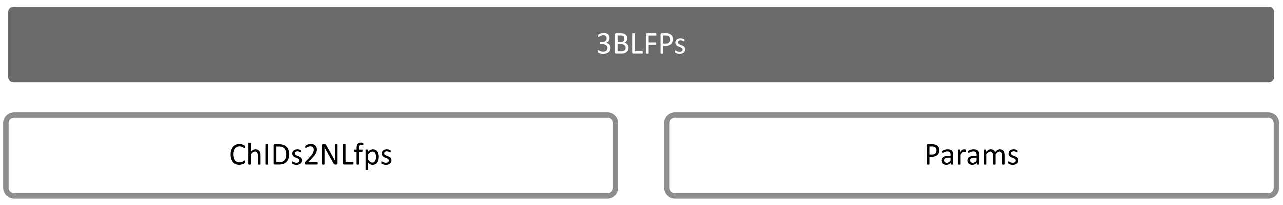
Contained objects

- ChIDs2NSpikes: a one-dimensional array to obtain the number of spikes found for each channel ID. The indexed position in the array corresponds to a channel ID and the contained value to the number of spikes identified for that channel. The information on the number of found spikes must match with the spikes effectively listed under the 3BChEvents result group
- Params: a scalar BwSpikeDetParams compound data type listing the parameters used for the spike detection. See paragraph 2.4.4. for details on the compound type
- FormParams: a scalar BwSpikeWaveformParams compound data type listing the parameters used for spike waveforms. See paragraph 2.4.5. for details on the compound type
- SortingParams: a scalar BwSpikeSortingParams compound data type listing the parameters used for the spike sorting. See paragraph 2.4.6. for details on the compound type
- BurtsParams: a scalar BwBurstDetParams compound data type listing the parameters used for the burst detection. Each compound is composed of:
    - MaxISIMs: a single floating-point value indicating the maximum distance in ms between to spikes in the same burst
    - MinNSpikes: an integer value indicating the minimum number of spikes required to define a burst

4.9. 3BLFPs

Minimum Version: 100
Current Version: 100

Version 100



Contained objects

- ChIDs2NLfps: a one-dimensional array to obtain the number of lfps found for each channel ID. The indexed position in the array corresponds to a channel ID and the contained value to the number of lfps identified for that channel. The information on the number of found lfps must match with the lfps effectively listed under the 3BChEvents result group
- Params: a scalar BwLfpDetParams compound data type listing the parameters used for the lfp detection. See paragraph 2.4.7. for details on the compound type

4.10. 3BEFPStimuli

Minimum Version: 100
Current Version: 100

Version	100

3BEFPStimuli

Params

Contained objects

- Params: a scalar BwEFPStimParams compound data type listing the parameters used for the efp stimulation. Each compound contains the information needed to define the stimulation used in the experiment

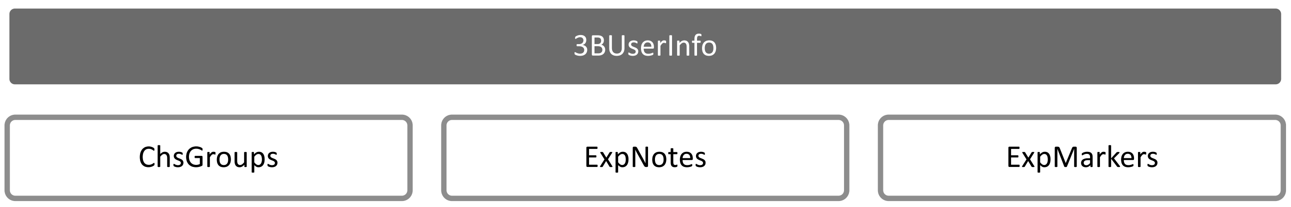
5. Shared Information

The Following information is present both in BRW-Files and in BXR-Files.

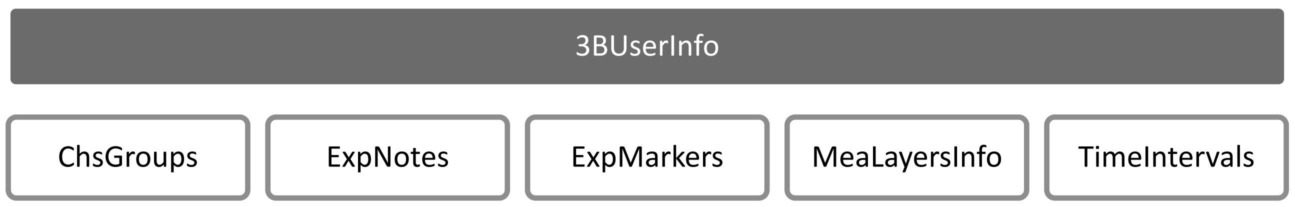
5.1. 3BUserInfo

Minimum Version: 100
Current Version: 102 in .brw, 103 in .bxr

Version 100 and 101



Version 102 and 103



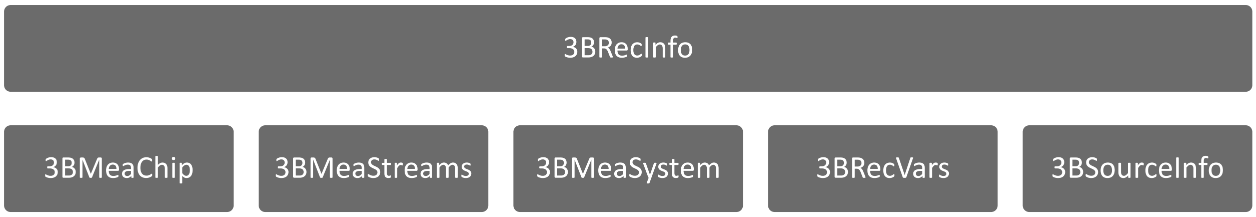
Contained objects

- ChsGroups: a one-dimensional array of a BwChsGroup compound data type (see paragraph 2.4.2)
- ExpNotes: an array of compound data types storing the user experimental notes. For version 100 each compound data type consists of a Title and a Value fields. For version 101 an additional FrameTimeStamp field is present indicating the time position within the data stream of the note
- ExpMarkers: a one-dimensional array of compound data type, storing the user experimental markers. Each compound consists of a Type, MarkIn, MarkOut, Description and Color fields
- MeaLayersInfo: a one-dimensional array of compound data type, storing information about the user-defined layers on the MEA
- TimeIntervals: a one-dimensional array of compound data type, storing information about the user-defined time intervals

5.2. 3BRecInfo

Minimum Version: 100
Current Version: 101 in .bxr, 102 in .brw

Version	from 100 to 102



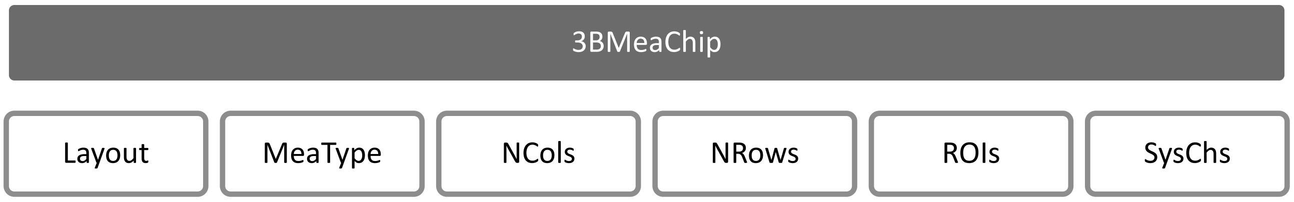
Contained objects

- 3BMeaChip: contains information on the MEA chip used during the recording
- 3BMeaStreams: lists the streams of data that have been recorded and their properties
- 3BMeaSystem: defines the system used to record data
- 3BRecVars: recording variables
- 3BSourceInfo [optional]: not present in case of a BRW-File; in case of a BXR-File it contains information on the source data file (i.e. on the linked BRW-File)

5.3. 3BMeaChip

Minimum Version: 100
Current Version: 101

Version 100 and 101



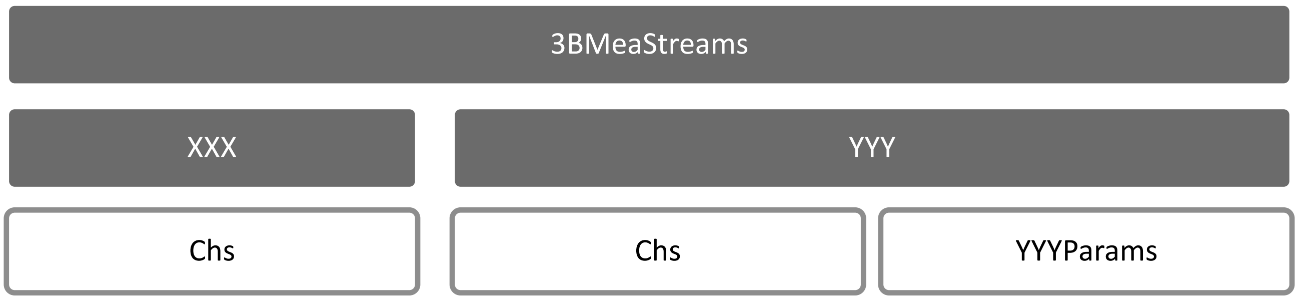
Contained objects

- Layout: a bi-dimensional array of Booleans of size N x M, where N is the number of rows of the MEA chip and M the number of columns. Each value in the layout array indicates whether the corresponding MEA chip channel (electrode) is present (true / 1) or not (false / 0)
- MeaType: a scalar value identifying the MEA type used to record data
- NRows: a scalar value indicating the number of rows of the MEA chip
- NCols: a scalar value indicating the number of columns of the MEA chip
- ROIs: a one-dimensional array of data elements representing the Region Of Interests (ROIs) settings applied for the recording. Each data element is a BwChsGroup compound data type
- SysChs: a one-dimensional array of Channels representing system channels, i.e. the channels that carry information either on the BioCAM system (e.g., calibration signals) or on the recorded protocol (e.g. external stimulus instants). Because of this SysChs does not contain for instance raw data information

5.4. 3BMeaStreams

Minimum Version: 100
Current Version: 101 in .bxr, 102 in .brw

Version from 100 to 102



Contained objects

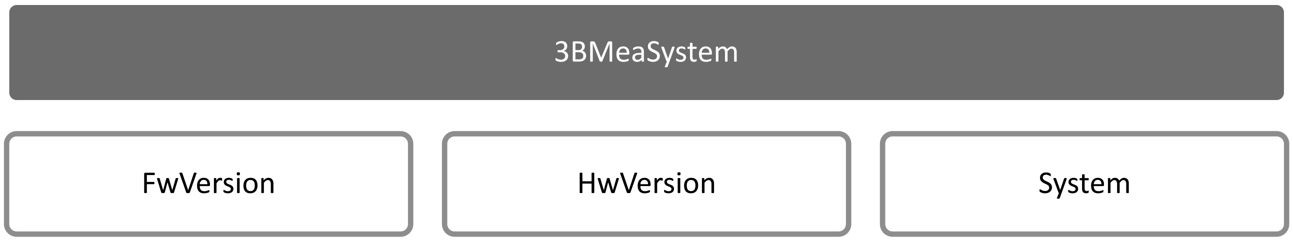
A group for each of the recorded MEA streams, named as the MEA stream. For instance, in case of raw data recording a Raw group. Each MEA group stream will then contain:

- Chs: a one-dimensional array of Channels listing the MEA channels that have been recorded for that stream
- YYYParams [optional]: one or more scalar compound data types listing the parameters used for the MEA stream. In case of Spike stream see 5.7 for details on the BwSpikeDetParams compound type

5.5. 3BMeaSystem

Minimum Version: 100
Current Version: 100

Version	100



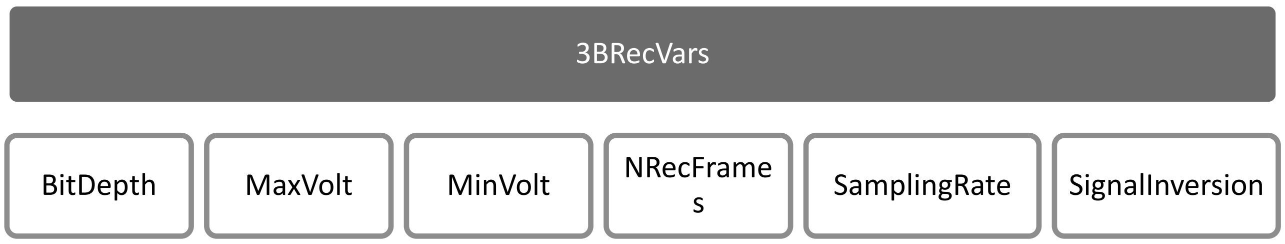
Contained objects

- FwVersion: the firmware Version of the system used to record the data whenever available
- HwVersion: the hardware Version of the system used to record the data whenever available
- System: a value identifying the system used to record the data

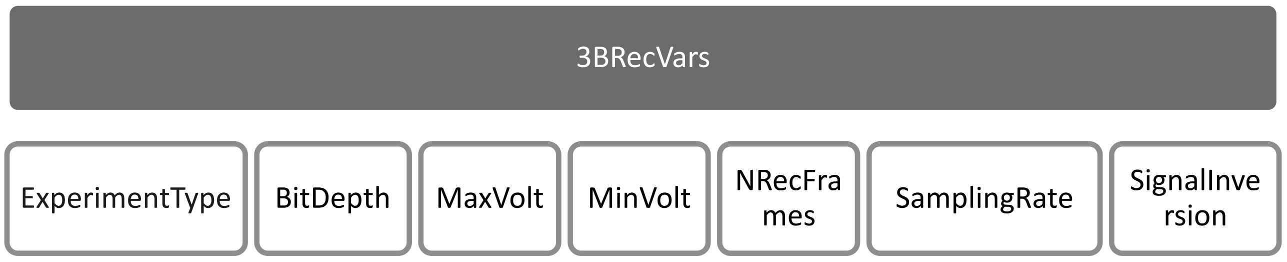
5.6. 3BRecVars

Minimum Version: 100
Current Version: 101

Version 100



Version 101



Contained objects

- BitDepth: an integer value indicating the bit depth for the sampled data (*)
- MaxVolt: a floating-point value for the maximum voltage of the recording amplifiers sampling range (*)
- MinVolt: a floating-point value for the minimum voltage of the recording amplifiers sampling range (*)
- NRecFrames: an integer value defining the number of recorded frames
- SamplingRate: a floating-point value for the sampling frequency used for recording data
- SignalInversion: either 1 or -1 to indicate whether the signal must be inverted or not (*)
- ExperimentType: an integer value indicating the experiment type, according to the following enumerator:

```
enum ExperimentType
{
   Standard = 0,
   EFP = 1
}
```

(*) These values taken together can be used to convert between digital sampled values and analog values in microVolt, according to the following formula:

```
AnalogValue = MVOffset + DigitalValue * ADCCountsToMV
```

Where `ADCCountsToMV` is defined as:

```
ADCCountsToMV = SignalInversion * ((MaxVolt - MinVolt)/2^BitDepth)
```

And `MVOffset` as:

```
MVOffset = SignalInversion * MinVolt
```

In order to convert a digital standard deviation value to an analog standard deviation value, the following formula may be used:

```
SDAnalog = ADdigital * StandardDeviationDAC
```

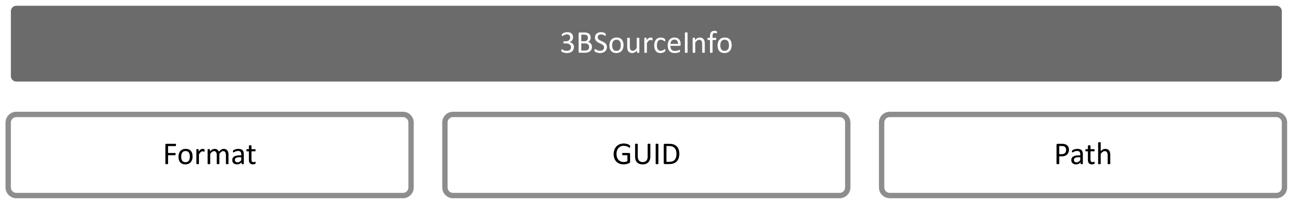
Where `StandardDeviationDAC` is defined as:

```
StandardDeviationDAC =(MaxVolt - MinVolt) / 2^BitDepth
```

5.7. 3BSourceInfo

Minimum Version: 100
Current Version: 100

Version 100



Contained objects

- Format: an integer value indicating the format of the source file. Currently the only accepted value is 0 for BRW-File.
- GUID: a global unique identifier for linking to the source file
- Path: the original path of the source file