.Shared Near Infrared File Format Specification
Last Updated – Aug 13, 2018
Copied Aug 7, 2018 from Aug 1, 2013

The file format specification uses the extension *.snirf. These are HDF5 format files, renamed with the .snirf extension. For a program to be "SNIRF-compliant", it must be able to read and write the SNIRF file.

The structure of each data file has a minimum of 5 required elements. All variables are double unless otherwise specified. All arrays are zero indexed.

"Standard" SNIRF file specification:

Required fields:

formatVersion - This is a string that specifies the version of the file format. This document describes format version "1.0"

data(idx) - This is a structure containing the data *d*, the time vector *t* of when the samples were acquired, and a description of the channels used to acquire the data. The data can be grouped in blocks indexed by *idx*. One convenient approach to blocking the data is to group all channels with the same time vector *t*.

data(idx).d - This is the actual raw data variable. This variable has dimensions of <number of time points> x <number of channels>. Columns in *d* are mapped to the measurement list (*ml* variable described below). The *d* variable can be complex (as in the case of sine-cosine demodulation for the laser carrier frequencies).

data(idx).t - The time variable. This provides the acquisition time of the measurement relative to the time origin. This will usually be a straight line with slope equal to the acquisition frequency, but does not need to be equal spacing. The size of this variable is <number of time points> x 1.

data(idx).ml - The measurement list. This variable serves to map the data array onto the probe geometry (sources and detectors), data type, and wavelength. This variable is an array structure that has the size <number

of channels> that describes the corresponding column in the data matrix.

For example, the *ml(3)* describes the third column of the data matrix (i.e. *d(:,3)*).

Each element of the array is a structure which describes the measurement conditions for this data with the following fields:

ml(chIdx).sourceIndex
ml(chIdx).detectorIndex
ml(chIdx).wavelengthIndex
ml(chIdx).dataType
ml(chIdx).dataTypeIndex

Optional fields include:

ml(chIdx).sourcePower
ml(chIdx).detectorGain

For example, if *ml(5)* is a structure with *sourceIndex*=2, *detectorIndex*=3, *wavelengthIndex*=1, *dataType*=1, *dataTypeIndex*=1 would imply that the data in the 5th column of the *d* variable was measured with source #2 and detector #3 at wavelength #1. Wavelengths (in nanometers) are described in the *sd.lambda* variable (described later). The data type in this case is 1, implying that it was a continuous wave measurement. The complete list of currently supported data types is found in the Appendix. The data type index specifies additional data type specific parameters that are further elaborated by other fields in the *sd* structure, as detailed below. Note that the Time Domain and Diffuse Correlation Spectroscopy data types have two additional parameters and so the data type index must be a vector with 2 elements that index the additional parameters.

sourcePower provides the option for information about the source power for that channel to be saved along with the data. The units are not defined, unless the user takes the option of using a *metaDataTag* described below to define, for

instance, *sourcePowerUnit*. *detectorGain* provides the option for information about the detector gain for that channel to be saved along with the data.

Note: The source indices generally refer to the optode naming (probe positions) and not necessarily the physical laser numbers on the instrument. The same is true for the detector indices. Each source optode would generally, but not necessarily, have 2 or more wavelengths (hence lasers) plugged into it in order to calculate deoxy- and oxy-hemoglobin concentrations. The data from these two wavelengths will be indexed by the same source, detector, and data type values, but have different wavelength indices. Using the same source index for lasers at the same location but with different wavelengths simplifies the bookkeeping for converting intensity measurements into concentration changes. As described below, optional variables *sd.srcLabels* and *sd.detLabels* are provided for indicating the instrument specific label for sources and detectors.

stim - This is an array describing any stimulus conditions. Each element of the array has the following required fields.

stim(n).name - This is a string describing the n^{th} stimulus condition.

stim(n).data - This is a three-column array specifying the stimulus time course for the n^{th} condition. Each row corresponds with a specific stimulus trial. The three columns indicate [starttime duration value]. The starttime, in seconds, is the time relative to the time origin when the stimulus takes on a value; the duration is the time in seconds that the stimulus value continues, and value is the stimulus amplitude. The number of rows is not constrained. (see examples in the appendix)

sd - This is a structured variable that describes the probe (source-detector) geometry. This variable has a number of required fields.

sd.lambda - This field describes the wavelengths used. This is indexed by the wavelength index of the *ml* variable.

For example, *sd.lambda* = [690 780 830]; implies that the measurements were

taken at three wavelengths (690nm, 780nm, and 830nm). The wavelength index of *ml(n).wavelengthIndex* variable refers to this field. *ml(n).wavelengthIndex* = 2 means the n^{th} measurement was at 780nm.

The number of wavelengths is not limited (except that at least two are needed to calculate the two forms of hemoglobin). Each source-detector pair would generally have measurements at all wavelengths.

sd.lambdaEmission - This field is required only for fluorescence data types, and describes the emission wavelengths used. The indexing of this variable is the same wavelength index in *ml* used for *sd.lambda* such that the excitation wavelength is paired with this emission wavelength for a given measurement.

sd.srcPos - This field describes the position (in *spatialUnit* units) of each source optode. This field has size <number of sources> x 3. For example, *sd.srcPos(1,:)* = [1.4 1 0], and *SpatialUnit*='cm'; places source number 1 at x=1.4 cm and y=1 cm and z=0 cm.

Dimensions are relative coordinates (i.e. to some arbitrary defined origin). The *qform* variable described below can be used to define the transformation between this SNIRF coordinate system and other coordinate systems.

sd.detPos - Same as *sd.srcPos*, but describing the detector positions.

There are additional required elements of the *sd* structure, depending on the data type of the measurement. These variables are indexed by *ml(chIdx).dataTypeIndex*:

Continuous wave (Fluorescence or non-fluorescence):
None

Frequency Domain (Fluorescence or non-fluorescence):

sd.frequency

Time domain – gated (Fluorescence or non-fluorescence):

sd.timeDelay
sd.timeDelayWidth

Time domain – moments (Fluorescence or non-fluorescence):

sd.momentOrder

Diffuse Correlation spectroscopy (Fluorescence or non-fluorescence):

sd.correlationTimeDelay
sd.correlationTimeDelayWidth

There are optional fields of the *sd* structure that can be used.

sd.srcLabels - This is a string array providing user friendly or instrument specific labels for each source. This can be of size <number of sources> x 1 or <number of sources> x <number of wavelengths>. This is indexed by *ml(chIdx).sourceIndex* and *ml(chIdx).wavelengthIndex*.

sd.detLabels - This is a string array providing user friendly or instrument specific labels for each detector. This is indexed by *ml(chIdx).detectorIndex*.

metaDataTags - This is a two column string array of arbitrary length consisting of any key/value pairs the user (or manufacturer) would like to put in. Each row of the array consists of two strings. Some possible examples:

['ManufacturerName','ISS'],
['Model','Imagent'],
['SubjectName', 'Pseudonym, I.M.A.'],
['DateOfBirth','20120401'],
['AcquisitionStartTime','150127.34']
['CalibrationFileName','phantomcal_121015.snirf']

While these tags are freeform, some conventions must be followed. Keys should use only alphanumeric characters with no spaces, with individual words capitalized. All values will be stored as strings, How strings are converted into numeric values is left to whoever defines the Key. However, it is required that dates be stored as YYYYMMDD, and clock times be stored as HHMMSS.SSSS… (24 hour format) for consistency. Time intervals must be in seconds.

The following metadata tags are required:

S*ubjectID*
MeasurementDate
MeasurementTime
SpatialUnit (allowed values are 'mm' and 'cm')

Optional variables:
These variables are not required for basic functions, but might be useful to get more out of your data sets.

aux- This optional array specifies any recorded auxiliary data. Each element of *aux* has the following required fields:

aux(n).name- This is string describing the n^{th} auxiliary data timecourse.

aux(n).d - This is the aux data variable. This variable has dimensions of <number of time points> x 1.

aux(n).t - The time variable. This provides the acquisition time of the aux measurement relative to the time origin. This will usually be a straight line with slope equal to the acquisition frequency, but does not need to be equal spacing. The size of this variable is <number of time points> x 1.

timeOffset – This variable specifies the offset of the file time origin relative to absolute (clock) time in seconds.

Appendix:

Supported data types for "d"

1 Raw - Continuous wave
2 Raw - Frequency Domain
3 Raw - Time domain - gated
4 Raw - Time domain – moments
5 Raw - Diffuse Correlation spectroscopy
6 Raw - Fluorescence – continuous wave

7 Raw - Fluorescence - Frequency Domain
8 Raw - Fluorescence - Time domain - gated
9 Raw - Fluorescence - Time domain – moments
10 Raw - Bioluminescence – continuous wave

Examples of stimulus waveforms

Assume there are 10 time points, starting at zero, spaced 0.1s apart. If we assume a stimulus to be a 0.2 second off, 0.2 second on repeating block, it would be specified as follows:
[0.2 0.2 1.0]
[0.6 0.2 1.0]

To Do:

Provide examples.

Further discuss meta data.

From the forum discussion:

Mathieu Coursolle and Blaise Frederick
In the optional variables, the "qform" variable specified a 4x4 matrix to align the NIRS coordinate system to other geometries. Would it be useful to have an additionnal variable that describes that geometry (ex: MNI, Talairach, anatomical, etc) ? I am thinking of something that may be similar to the NIfTI file format.

That sounds like a good idea; we'll have to think about what geometries would make the most sense ("scanner anatomic" is probably not relevant, but "MNI", "Talairach", maybe "10-20" might be good choices- we'd have to decide how the last would be implemented).

qform- This variable specifies a 4x4 qform matrix to align the NIRS coordinate system to other geometries.

Alex Cristia

- You could add a field sd.Origin which can specify the 10-20 electrode used as reference, since most fNIRS neurocog users will have one. This simple addition would make it much easier to incorporate localization in an eventual meta-analysis.

David Boas

One issue that remains to be resolved is how to handle calculated or derived data types. The specification presently supports several raw data types. It is desirable to add a data type for concentration results. An issue we are struggling with is that every channel of data, i.e. column of "d", has a corresponding descriptor in the "ml" structure. The "ml" structure indexes the source, detector, and data type for the corresponding data channel. It also indexes the wavelength. For concentration, there is no wavelength. Thus, it seems that if we have a data type for concentration, then the corresponding "ml(n).wavelengthIndex" field would be ignored. In addition, the "ml(n).dataTypeIndex" could be used to reference what chromophore is stored in the data. The list of chromophores could be provided by sd.Chromophores, which could be a string array with possible entries of "HbO", "HbR", "H2O", "aa3", etc.