Neuroshare Native File Format Specification Rev 0.9d

Standard file format for storing classified neurophysiology data

Aug 2003

AFFILIATIONS

This standard is being developed and maintained through the Neuroshare Project. The purpose of this project is to create open, standardized methods for accessing neurophysiological experiment data from a variety of different data formats, as well as open-source software tools based on these methods. All standards and software resulting from the Neuroshare Project are distributed and revised through the http://www.neuroshare.org web site. Additional contact information and project history can also be accessed through this site.

DISCLAIMER

This specification document is provided "as is" with no warranties whatsoever, including any warranty of merchantability, non-infringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification or sample. The Neuroshare project and the working group disclaim all liability relating to the use of information in this specification.

TRADEMARKS

Windows and Microsoft are registered trademarks of the Microsoft Corporation. All other product names are trademarks or service marks of their respective owners.

REVISIONS

This is the first draft release of the Neuroshare Native File Format Specification version 0.9.

COPYRIGHT AND DISTRIBUTION

Contents

Introduction

The Neuroshare Native (NsN) Data File specifications define a standard file format for storing common types of neurophysiology experiment data. It is primarily intended for saving classified neural waveform information and providing this information in an easily accessible format for other analysis software packages. The size of the Neuroshare Native Data File can be significantly reduced from the original raw experimental data file, because classified unit events are redefined as Neural Event Entities and only the timestamps are saved.

The structure of the information headers and the data formats are based on the Neuroshare API Library specifications. Entity data is saved so that the informational header is immediately followed by a block of binary data values. This grouped information constitutes a data element and is identified by a tag structure, which specifies the type and the length in bytes of the following information. Reading this data becomes very simple. It requires a parser to scan each data element and dispatch the data element to the proper access procedure.

Overview of Data Elements

The *.nsn data format consists of several types of tagged data elements. The general structure for one data entity is a tag, followed by one or more informational header structures and then a block of binary data. The only information that is not tagged is the general file information structure placed at the beginning of the data file.

1. Each of the four Neuroshare entity types are represented with the following format:

   a). General Entity Information Header

   This provides general information about the types, number of the entity specific headers and data types saved in the data file and is stored in the Neuroshare API structure ns_ENTITYINFO.

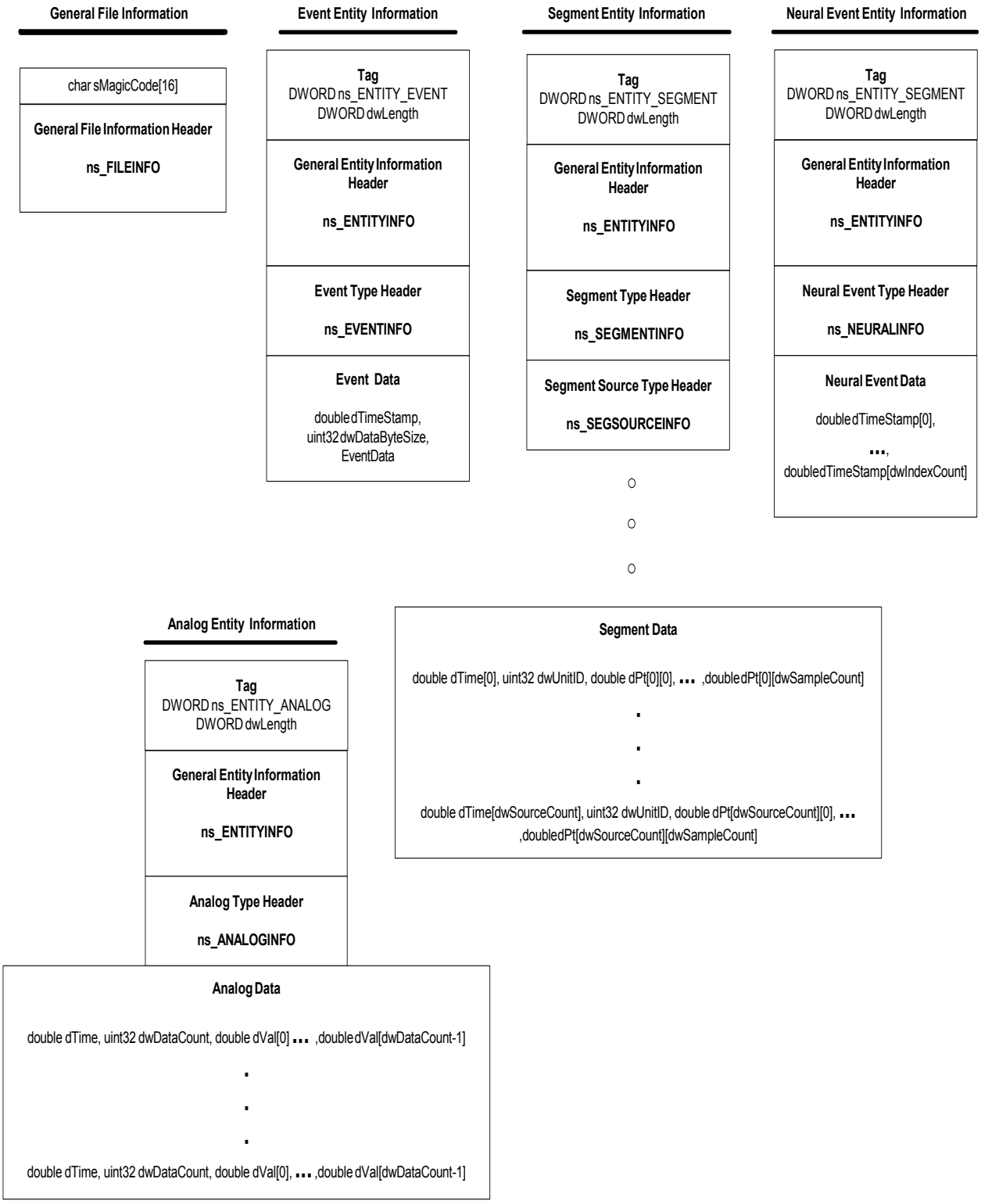
   b). Entity Specific Information Headers

   This consists of a variable number of entity specific information headers that describe in more detail the blocks of data that follow. This is comprised of the specific entity information structures defined in the Neuroshare API.

   c). Data Blocks

   Binary data for the entity type.

FIgure 1. Neuroshare Native File Data Elements Overview



Data Conventions

The following primitive data types are used here and are defined as follows:

`char`	8-bit character value normally reserved for ASCII strings
`int8`	8-bit (1 byte) signed integers
`uint8`	8-bit (1 byte) unsigned integers
`int16`	16-bit (2 byte) signed integers
`uint16`	16-bit (2 byte) unsigned integers
`int32`	32-bit (4 byte) signed integers
`uint32`	32-bit (4 byte) unsigned integers
`double`	64-bit, double precision floating point value

Multi-byte data is stored in little endian format so that the low-order byte of the number is stored first and the high-order byte last (LSB first, MSB last). All analog values in this library, including time, shall use a 64-bit double-precision floating point representation. All analog entities also include a text field for reporting measurement units such as “meters”, “MPa”, “kg”. The use of metric units is strongly encouraged. Time is always reported in seconds.

Parameters, structures and their members are all aligned to 32-bits.

Classification Unit Definitions

Classified neural events are saved in the dwUnitID field for segment and neural data. This field is defined as a bit-field and supports multiple classification codes. The values are defined as follows:

0x0000	unclassified
0x0001	noise or artifact
0x0002	unit 1 present
0x0004	unit 2 present, etc.

Tag Identifiers

Elements in the data file are preceded by a structure containing a type flag, denoting the type of data to follow and the length in bytes of the following block of data and information headers. This structure is defined as:

ns_TAGELEMENT

```
typedef struct
{
        uint32 dwElemType;          // Type identifier describing the type of data or information that
                                    // follows.
        uint32 dwElemLength;        // Length in bytes of the following data.

} ns_TAGELEMENT;
```

The defined type identifiers are:

```
const uint32 ns_ENTITY_UNKNOWN      =     0;     // Unknown data entity type
const uint32 ns_ENTITY_EVENT        =     1;     // Event entity type
const uint32 ns_ENTITY_ANALOG       =     2;     // Analog entity type
const uint32 ns_ENTITY_SEGMENT      =     3;     // Segment entity type
const uint32 ns_ENTITY_NEURAL       =     4;     // Neural event entity type
const uint32 ns_INFO_FILE           =     5;     // File information
```

The "ns_ENTITY_*xxxxxx*" type identifiers are the standard entity type identifiers defined by the Neuroshare API specifications. Additional tag types may defined for other data elements in the future.

General File Information

The beginning of the data file contains one field identifying the data file type and the version number. Immediately following is the structure containing general file information.

char sMagicCode[16]

Document type identifier equal to "NSN ver000000010" specifying a Neuroshare Native Data file that complies with ver 1.0 of the Neuroshare Native Datafile Specification.

File Information Header

The ns_ FILEINFO structure, as defined by the Neuroshare API specification, provides general information about the data file entity type.

ns_FILEINFO

```
typedef struct {
    char szFileType[32];            // Human readable file type descriptor.
    uint32 dwEntityCount;           // Number of entities in the data file.  This number is used
                                    // to enumerate all the entities in the data file from 0 to
                                    // (dwEntityCount –1) and to identify each entity in
                                    // function calls (dwEntityID).
    double dTimeStampResolution     // Minimum timestamp resolution in seconds.
    double dTimeSpan;               // Time span covered by the data file in seconds.
    char szAppName[64];             // Information about the application that created the file.
    uint32 dwTime_Year;             // Year.
    uint32 dwTime_Month;            // Month (1-12; January = 1).
    uint32 dwTime_DayOfWeek;        // Current day of the week (Sunday = 0)
    uint32 dwTime_Day;              // Day of the month (1-31).
    uint32 dwTime_Hour;             // Hour since midnight (0-23).
    uint32 dwTime_Min;              // Minute after the hour (0-59).
    uint32 dwTime_Sec;              // Seconds after the minute (0-59).
    uint32 dwTime_MilliSec;         // Milliseconds after the second (0-1000).
    char szFileComment[256];        // Extended comments
} ns_FILEINFO;
```

Event Entity Element

This element contains information on event entity data. It consists of the following fields:

1. Element Tag

Information identifying the type and size of the following data.

ns_TAGELEMENT

```
typedef struct{
        uint32 dwElemType;          // Element type ns_ENTITY_EVENT
        uint32 dwElemLength;        // Size in bytes of the following data block
} ns_TAGELEMENT;
```

2. General Entity Information Header

The ns_ENTITYINFO structure, as defined by the Neuroshare API specification, specifies the label, the entity type and the item count of the entity saved.

ns_ENTITYINFO

```
typedef struct {
    char szEntityLabel[32];         // Specifies the label or name of the entity.
    uint32 dwEntityType;            // Flag specifying the type of entity data saved.
    uint32 dwItemCount;             // Number of data items for the specified entity in the file.
} ns_ENTITYINFO;
```

3. Event Specific Information Header

This header provides specific information about the event entity saved, the label, the data type of the event entity, and the number of items. The ns_EVENTINFO structure is defined in the Neuroshare API Specifications

ns_EVENTINFO.

```
typedef struct {
    uint32 dwEventType;          // A type code describing the type of event data associated with
                                 // each indexed entry.  The following information types are
                                 // allowed:
                                 // #define ns_EVENT_TEXT      0   //text string
                                 // #define ns_EVENT_CSV       1   //comma separated values
                                 // #define ns_EVENT_BYTE      2   // 8-bit binary values
                                 // #define ns_EVENT_WORD      3   //16-bit binary values
                                 // #define ns_EVENT_DWORD     4   //32-bit binary values
  uint32 dwMinDataLength;        // Minimum number of bytes that can be returned for an Event.
  uint32 dwMaxDataLength;        // Maximum number of bytes that can be returned for an Event.
  char szCSVDesc [128];          // Provides descriptions of the data fields for CSV Event Entities.
} ns_EVENTINFO;
```

4. Event Data

An individual event data item is saved with the following format:

```
    double dTimestamp          // Time of the event in seconds.
    uint32 dwDataByteSize      // Size in bytes of the following data value.
    EventValue                 // Binary data values of events.  The type of this value is
                               // specified by  ns_EVENTINFO.dwEntityType.
```

Analog Entity Element

This element contains information on Analog entity data. It consists of the following fields:

1. Element Tag

ns_TAGELEMENT
Information identifying the type and size of the following data. The structure is defined above.

2. General Entity Information Header

This consists of the following two fields:

ns_ENTITYINFO

This structure is defined above. The parameter *dwItemCount* specifies the total number of analog data values saved for this entity.

3. Analog Entity Specific Information Header

This header provides specific information about the analog entity saved, such as, the sampling rate, the minimum and maximum values of the input signal, the units of measurement, location and filtering characteristics. Note, there are no accommodations for taking into account unequal time gaps between samples. The ns_ANALOGINFO structure is defined by the Neuroshare API specification.

ns_ANALOGINFO

```
typedef struct{
    double dSampleRate;         // The sampling rate in Hz used to digitize the analog values.
    double dMinVal;             // Minimum possible value of the input signal.
    double dMaxVal;             // Maximum possible value of the input signal.
    char szUnits[16];           // Specifies the recording units of measurement.
    double dResolution;         // Minimum input step size that can be resolved.
                                // (E.g. for a +/- 1 Volt 16-bit ADC this value is .0000305).
    double dLocationX;          // X coordinate of source in meters.
    double dLocationY;          // Y coordinate of source in meters.
    double dLocationZ;          // Z coordinate of source in meters.
    double dLocationUser;       // Additional manufacturer-specific position information
                                // (e.g. electrode number in a tetrode).
    double dHighFreqCorner;     // High frequency cutoff in Hz of the source signal filtering.
```

```
    uint32 dwHighFreqOrder;         // Order of the filter used for high frequency cutoff.
    char szHighFilterType[16];      // Type of filter used for high frequency cutoff (text format).
    double dLowFreqCorner;          // Low frequency cutoff in Hz of the source signal filtering.
    uint32 dwLowFreqOrder;          // Order of the filter used for low frequency cutoff.
    char szLowFilterType[16];       // Type of filter used for low frequency cutoff (text format)..
    char szProbeInfo[128];          // Additional text information about the signal source.
} ns_ANALOGINFO;
```

4. Analog Data

Consecutively recorded analog data is saved with a start time, the size in bytes of the analog data to follow and a series of double values. The first value occurs at the time specified by dTimestamp and consecutive values occur at 1/(ns_ANALOGINFO.dSampleRate) seconds apart. If there is a jump in the sample period, another group of data consisting of dStartTime, dwDataCount and the analog data values is appended. The total number of analog data values **dwDataCount[0] + … + dwDataCount[n]** is equal to dwIndexCount.

For example:

double dTimestamp, uint32 dwDataCount[0], double dAnalogValue[0], … ,
double dAnalogValue[dwDataCount[0]-1]
double dTimestamp, uint32 dwDataCount[1], double dAnalogValue[0], … ,
double dAnalogValue[dwDataCount[1]-1]

.
.
.

Segment Entity Element

This element contains information on Segment entity data. It consists of the following fields:

1. Element Tag

ns_TAGELEMENT
Information identifying the type and size of the following data. The structure is defined above.

2. General Entity Information Header

This consists of the following two fields:

ns_ENTITYINFO

This structure is defined above.

3. Segment Entity Specific Information Header

Segment entity information consists of one information header and a variable number of source information headers. The primary header provides specific information about the number of sources from which the data was recorded from, the number of samples in each waveform, the sampling rate and the units of measurement

The primary information header consists of the following structure:

ns_SEGMENTINFO

This is the same structure defined in the Neuroshare API Specifications.

```
typedef struct {
    uint32 dwSourceCount;        // Number of sources contributing to the Segment Entity data.
    uint32 dwMinSampleCount;     // Minimum number of samples in each Segment data item.
    uint32 dwMaxSampleCount;     // Maximum number of samples in each Segment data item.
    double dSampleRate;          // The sampling rate in Hz used to digitize source signals.
    char szUnits[32];            // Specifies the recording units of measurement.
} ns_SEGMENTINFO
```

4. Segment Source Entity Specific Information Header

The number (specified by ns_SEGMENTINFO.dwSourceCount) of source information headers, following the primary header, depends on the number of sources contributing to the data. The source information header contains information about each of the contributing sources of the segment data. It specifies the minimum and maximum values of the input signal, input resolution, time shift from the recorded timestamp, location and filtering characteristics.

The segment source information header has the following structure:

ns_SEGSOURCEINFO

This is the same structure defined in the Neuroshare API Specifications.

```
typedef struct {
    double dMinVal;             // Minimum possible value of the input signal.
    double dMaxVal;             // Maximum possible value of the input signal.
    double dResolution;         // Minimum input step size that can be resolved.
                                // (E.g. for a +/- 1 Volt 16-bit ADC this value is .0000305).
    double dSubSampleShift;     // Time difference (in sec) between the nominal timestamp
                                // and the actual sampling time of the source probe. This
                                // value will be zero when all source probes are sampled
                                // simultaneously.
    double dLocationX;          // X coordinate of source in meters.
    double dLocationY;          // Y coordinate of source in meters.
    double dLocationZ;          // Z coordinate of source in meters.
    double dLocationUser;       // Additional manufacturer-specific position information
                                // (e.g. electrode number in a tetrode).
    double dHighFreqCorner;     // High frequency cutoff in Hz of the source signal filtering.
    uint32 dwHighFreqOrder;     // Order of the filter used for high frequency cutoff.
    char szHighFilterType[16];  // Type of filter used for high frequency cutoff (text format).
    double dLowFreqCorner;      // Low frequency cutoff in Hz of the source signal filtering.
    uint32 dwLowFreqOrder;      // Order of the filter used for low frequency cutoff.
    char szLowFilterType[16];   // Type of filter used for low frequency cutoff (text format)..
    char szProbeInfo[128];      // Additional text information about the signal source.
} ns_SEGSOURCEINFO;
```

5. Segment Data

Segment data is saved as a 2-dimensional array with *ns_SEGMENTINFO.dwSourceCount* number of rows containing *ns_SEGMENTINFO.dwMaxSampleCount* number of double values in each row. Each row represents a waveform and starts with a timestamp and a unit definition. The classification unit ID field is defined above.

Segment data is saved in the following format:

double dTimestamp, uint32 dwUnitID, double dValue[0][0], double dValue[0][1], …,
double Value[0][dwSampleCount]
.
.
.
double dTimestamp, uint32 dwUnitID, double dValue[dwSourceCount][0],
double dValue[dwSourceCount][1], …,
double dValue[dwSourceCount][dwSampleCount]

Neural Event Entity Element

This element contains information on neural event entity data. It consists of the following fields:

1. Element Tag

ns_TAGELEMENT
Information identifying the type and size of the following data. The structure is defined above.

2. General Entity Information Header

This consists of the following two fields:

ns_ENTITYINFO

This structure is defined above.

3. Neural Event Entity Specific Information Header

This header provides specific information about the neural entity saved, such as, the source entity ID, the unit definition assigned to it and text information about the recording probe. ns_NEURALINFO is defined in the Neuroshare API Specifications.

ns_NEURALINFO

```
typedef struct {
    uint32 dwSourceEntityID;     // Optional ID number of a source entity.  If the Neural Event is
                                 // derived from other entity sources, such as Segment Entities,
                                 // this value links the Neural Event back to the source.
    uint32 dwSourceUnitID;       // Optional sorted unit ID number used in the source Entity.
    char szProbeInfo[128];       // Text information about the source probe or the label of a
                                 // source Segment Entity.
} ns_NEURALINFO
```

4. Neural Event Data

Classified spike waveforms, derived from either segment or analog entities and originating from the "same" neural source, are created into neural event entities. The classification unit ID field is defined above and is specified in the neural event specific information header. The waveforms are no longer retained, as neural events are saved as a series of timestamps marking the occurrence of a signal. Neural event data has the following format:

double dTimestamp[0], … , double dTimestamp[dwItemCount]

Revisions

Ver 0.9d

- ns_FileInfo structure added dwTime_DayofWeek parameter. p 9.
- ns_FileInfo enumeration of month starts at 1, i.e. January = 1. p9
- removed dwEntityDataOffset from all data element definitions
- Tag is removed from FILEINFO structure at beginning of data file.