ns_CreateFile Class Library Manual
Ver 1.3
2011/03/04

(English Translation 2010/07/30)
Contents

INTRODUCTION .................................................................................................................................................. 3

OBJECTIVE OF THE NS_CREATEFILE CLASS LIBRARY ......................................................................... 4

METHODS SUMMARY ..................................................................................................................................... 5

USEFUL CONSTANTS ...................................................................................................................................... 7

NEUROSHARE FILE CREATION FLOW ......................................................................................................... 9

SHARED FUNCTIONS ..................................................................................................................................... 11

OBJECT MANIPULATION PROHIBITION FUNCTION ................................................................................... 11

STRUCTURE AUTOMATIC UPDATE FUNCTION ......................................................................................... 12

INPUT VALUE REVISION FUNCTION[DOUBLE NATURAL NUMBER—UINT32] ............................................. 15

INPUT VALUE REVISION FUNCTION[MODIFY LENGTH OF CHARACTER] .................................................. 15

ERROR FUNCTION .......................................................................................................................................... 16

WARNING FUNCTION .................................................................................................................................... 17

METHOD DETAILS ....................................................................................................................................... 18

NEUROSHARE FILE CREATION PROCESSING .............................................................................................. 18

EVENT ENTITY REGISTRATION PROCESSING .............................................................................................. 25

ANALOG ENTITY REGISTRATION PROCESSING ............................................................................................ 32

SEGMENT ENTITY REGISTRATION PROCESSING .......................................................................................... 39

NEURAL EVENT ENTITY REGISTRATION PROCESSING .............................................................................. 50

NEURAL EVENT ENTITY REGISTRATION PROCESSING .............................................................................. 50

TERMINOLOGY SUPPLEMENT.................................................................................................................... 57

NSOBJ ............................................................................................................................................................... 57

NSA_***INFO ................................................................................................................................................. 57

CHART SUPPLEMENT .................................................................................................................................... 61

NOTICE POINTS................................................................................................................................................. 62

VER 1.0 .............................................................................................................................................................. 62

VER 1.3 .............................................................................................................................................................. 62

UPDATE LOG ....................................................................................................................................................... 63

ACKNOWLEDGMENT .................................................................................................................................... 63
Introduction

This handbook is the explanatory material for the ns_CreateFile class library. We recommend that you read this handbook after you have grasped the organization of the Neuroshare format (1).

The ns_CreateFile class library is a group of functions (hereafter “Methods”) which convert standard neurophysiological experiment data into normalized Neuroshare format data without inconsistencies. The ns_CreateFile class library can be run in Windows XP SP2 and the MATLAB 7.5 (R2007b) environment.

Moreover, for examples of data conversion using the ns_CreateFile class library on dummy data, and applications, see the m-file SampleConverter.m. For examples of concrete uses for the ns_CreateFile class library, please refer here.

(1) Neuroshare Format: Neurophysiological data-sharing format. Extension (.nsn). Represents neurophysiological experimental data in terms of four defined Entity types, and a combination of structure and Entity data. Please refer to <Figure 1 Neuroshare Format> and the document “Neuroshare Native File Format Specification Rev 0.9d”. Neuroshare site: https://neuroshare.sourceforge.net/download.shtml.
Objective of the ns_CreateFile class library

This is an introduction to the objective of the ns_CreateFile class library.

The goal of the ns_CreateFile class library is
“to convert typical neurophysiological experiment data (henceforth “experiment data”), into standardized Neuroshare format data.”

Already, the ns_CreateFile class library contains functions adequate to create standardized Neuroshare data.

In this handbook, the ns_CreateFile class library methods, usable constants, Neuroshare file production procedure, file-sharing functions for various methods, and the order of specialized functions for individual methods are explained.

(1) “Conversion to correct Neuroshare format data.”
According to the ns_CreateFile class library, a file originating from experimental data is “converted to correct Neuroshare format data” when it meets the following conditions:

1. All contents recorded in the file’s general structure are compatible with the definition of Neuroshare format.
2. All contents recorded in the file’s Entity data are compatible with the definition of Neuroshare format.
3. There is no disparity between the file’s internal structure and its Entity data.
Methods Summary

The ns_CreateFile class library includes the following methods. Method names and function summaries are stated here. For detailed functions for each method, please see <Method Details>.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Function Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_CreateFile</td>
<td>Create a Neuroshare data storage object (&quot;nsObj&quot;※1)</td>
</tr>
<tr>
<td>ns_GetFileInfo</td>
<td>Retrieve file information structure (nsa_FILEINFO ※2)</td>
</tr>
<tr>
<td>ns_SetFileInfo</td>
<td>Modify nsa_FILEINFO (※2)</td>
</tr>
<tr>
<td>ns_CloseFile</td>
<td>Integrate all data</td>
</tr>
<tr>
<td>ns_NewEventData</td>
<td>Generate new event data</td>
</tr>
<tr>
<td>ns_GetEventInfo</td>
<td>Retrieve Event information structure (nsa_EVENTINFO ※2)</td>
</tr>
<tr>
<td>ns_SetEventInfo</td>
<td>Modify nsa_EventInfo (※2)</td>
</tr>
<tr>
<td>ns_AddEventInfo</td>
<td>Add event data</td>
</tr>
<tr>
<td>ns_NewAnalogData</td>
<td>Generate new analog data</td>
</tr>
<tr>
<td>ns_GetAnalogInfo</td>
<td>Retrieve analog information structure (nsa_ANALOGINFO ※2)</td>
</tr>
<tr>
<td>ns_SetAnalogInfo</td>
<td>Modify nsa_ANALOGINFO (※2)</td>
</tr>
<tr>
<td>ns_AddAnalogData</td>
<td>Add analog data</td>
</tr>
</tbody>
</table>

<Segment Entity Registration Processing>

ns_NewSegmentData          Generate new segment data
ns_GetSegmentInfo Retrieve Segment information structure
    (nsa_SEGMENTINFO(×2))

ns_GetSegmentSourceInfo Retrieve segment source information structure
    (nsa_SEGSOURCEINFO(×2))

ns_SetSegmentInfo Modify nsa_SEGMENTINFO(×2)

ns_SetSegmentSourceInfo Modify nsa_SEGSOURCEINFO(×2)

ns_AddSegmentData Add ns_SEGSOURCEINFO or segment data

<Neural Event Entity Registration Processing>
ns_NewNeuralEventData Generate new neural event data

ns_GetNeuralInfo Retrieve neural event information structure
    (nsa_NEURALINFO(×2))

ns_SetNeuralInfo Modify nsa_NEURALINFO(×2)

ns_AddNeuralEventData Add neural event data

<Other Functions>
display Display present value of a structure member
    (Displays only locations according to the user’s configuration)

display_All_Information Display each structure member’s current value
    (Displays all member variables)

subsasgn (Dot) Prohibit modification of each structure member
    variable which utilizes an operator

(×1) An nsObj is “an object which stores Neuroshare format data.” For details, see <nsObj>
(×2) An nsa_***INFO structure is “a structure containing member variables from an ns_***INFO structure
    which are not consistent with Entity Data format. For details, see <nsa_***INFO>
Useful Constants

Constant values used with the ns_CreateFile class library are indicated here.

Constants used with the ns_CreateFile class library are listed in the following <Table 1 Useful Constants >

The new constants used in the ns_CreateFile class library are added to the constants defined in Neuroshare Matlab API Specification Rev2.2(*1)

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Actual Value</th>
<th>Meaning of the Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_OK</td>
<td>0</td>
<td>Normal termination of method processing (Includes WARNING processing time, &lt;See &quot;WARNING processing function&quot;)</td>
</tr>
<tr>
<td>ns_LIBERROR</td>
<td>-1</td>
<td>Library link error</td>
</tr>
<tr>
<td>ns_TYPEERROR</td>
<td>-2</td>
<td>File opening error</td>
</tr>
<tr>
<td>ns_FILEERROR</td>
<td>-3</td>
<td>File access error</td>
</tr>
<tr>
<td>ns_BADFILE</td>
<td>-4</td>
<td>File handle error</td>
</tr>
<tr>
<td>ns_BADENTITY</td>
<td>-5</td>
<td>EntityID description error</td>
</tr>
<tr>
<td>ns_BADSOURCE</td>
<td>-6</td>
<td>SourceID description error</td>
</tr>
<tr>
<td>ns_BADINDEX</td>
<td>-7</td>
<td>Index error</td>
</tr>
<tr>
<td>ns_WRONGLABEL</td>
<td>-101</td>
<td>Input argument character string specification error (Error in the EntityLabel specification indicating a file name)</td>
</tr>
<tr>
<td>ns_WRONGID</td>
<td>-102</td>
<td>Error in the SegmentSourceID specification for an input argument’s EntityID</td>
</tr>
<tr>
<td>ns_WRONGHEADER</td>
<td>-103</td>
<td>Error in the content of an input argument’s structure</td>
</tr>
<tr>
<td>ns_WRONGDATA</td>
<td>-104</td>
<td>Error in the content of an input argument’s data</td>
</tr>
</tbody>
</table>

< Set values for member variables: ns_TATELEMENT ns_ENTITYINFO >

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Actual Value</th>
<th>Meaning of the Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_ENTITY_UNKOWN</td>
<td>0</td>
<td>Entity type ambiguity</td>
</tr>
<tr>
<td>ns_ENTITY_EVENT</td>
<td>1</td>
<td>Event Entity</td>
</tr>
<tr>
<td>ns_ENTITY_ANALOG</td>
<td>2</td>
<td>Analog Entity</td>
</tr>
<tr>
<td>ns_ENTITY_SEGMENT</td>
<td>3</td>
<td>Segment Entity</td>
</tr>
<tr>
<td>ns_ENTITY_UNKOWN</td>
<td>4</td>
<td>Neural Event Entity</td>
</tr>
<tr>
<td>ns_INFO_FILE</td>
<td>5</td>
<td>File Information</td>
</tr>
</tbody>
</table>

< Set values for member variables: ns_EVENTINFO >

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Actual Value</th>
<th>Meaning of the Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_EVENT_TEXT</td>
<td>0</td>
<td>Character string</td>
</tr>
<tr>
<td>ns_EVENT_CSV</td>
<td>1</td>
<td>Comma-delimited values</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>ns_EVENT_BYTE</td>
<td>2</td>
<td>8-bit binary values</td>
</tr>
<tr>
<td>ns_EVENT_WORD</td>
<td>3</td>
<td>16-bit binary values</td>
</tr>
<tr>
<td>ns_EVENT_DWORD</td>
<td>4</td>
<td>32-bit binary values</td>
</tr>
</tbody>
</table>

(※1) Neuroshare Matlab API Specification Rev2.2: A library which reads Neuroshare format files
Please see Neuroshare Matlab API Specification Rev2.2.
Neuroshare site: http://neuroshare.sourceforge.net/download.shtml
Neuroshare File Creation Flow

The following is the flow for creating a Neuroshare-format file

In a MATLAB Workspace user job, methods are arranged in a use sequence and sorted into the following 1.-10. categories (Underlined words are the method names)

After the following steps are executed, various features of nsObj can not be changed.

Please see <Figure 2 Sample Data Creation Flow[1]>,<Figure 3 Sample Data Creation Flow[2]>, and <Figure 4 Sample Data Creation Flow[3]>. For example, the place where event data is registered. ( >> Input example in a Work Space)

1. **ns_CreateFile** ・・・ Create a Neuroshare format file
   >> [retA nsObj] = ns_CreateFile( 'sample.nsn' );
2. **ns_GetFileInfo** ・・・ Retrieve an nsa_FILEINFO file
   >> [retB nsa_FILEINFO] = ns_GetFileInfo( nsObj );
3. **[Modification of structure contents dependent on the user]**
   >> nsa_FILEINFO.szFileName = 'Sample';
4. **ns_SetFileInfo** ・・・ Update nsa_FILEINFO
   >> [retC nsObj] = ns_SetFileInfo( nsObj, nsa.FILEINFO );
5. **ns_NewEventData** ・・・ Generate new event data process
   >> [retD nsObj EventID] = ns_NewEventData( nsObj, 'dummy' );
6. **ns_GetEventInfo** ・・・ Retrieve nsa_EVENTINFO
   >> [retE nsa_EVENTINFO] = ns_GetEventInfo( nsObj, EventID );
7. **[Modification of structure contents dependent on the user]**
   >> nsa_EVENTINFO.szCSVDesc = 'Words empty of meaning';
8. **ns_SetEventInfo** ・・・ Update nsa_EVENTINFO
   >> [retF nsObj] = ns_SetEventInfo( nsObj, EventID, nsa_EVENTINFO );
9. **ns_AddEventData** ・・・ Add to Event Info
   >> [retG nsObj] = ns_AddEventData( nsObj, EventID, 1.5, uint32( 23 ) );
10. **ns_CloseFile** ・・・ Integrate all data
    >> [retH] = ns_CloseFile( nsObj );

※ If event data for the same EntityID is registered multiple times in a location, repeat step 9
※ For a location in which multiple Event IDs are registered, repeat steps 5-9 a second time.
※ Perform step 10 when finished registering all data.
1. ns_CreateFile()

2. ns_GetFileInfo()

3. Update values.

4. ns_SetFileInfo()

5. ns_NewEventData()

6. ns_GetEventInfo()

7. Update values

8. ns_SetEventInfo()

9. ns_AddEventData()

"filename".nsn (Neuroshare Format File)

10. ns_CloseFile()
Shared Functions

The ns_CreateFile class library sharing functions are described here.

Object Manipulation Prohibition Function

This function makes it impossible to directly modify the contents of an nsObj, an object in the ns_Create File class library (in other words, experiment data), except using a class library function.

This function prevents the contents of an nsObj in the Work Space from being directly manipulated, and prevents values which depart from Neuroshare format to be stored in an nsObj.

Furthermore, when a request is made to directly manipulate the contents of an nsObj, the following messages are output in order to prohibit the request.

[Example of the object manipulation prohibition function]
When a request is made to directly manipulate an ns_FILEINFO member variable for a given nsObj:
<< Input to Work Space >> nsObj.ns_FILEINFO.szFileType = 'meaningless words';

【Operation Result】
Message: ERROR: Can not edit a member of [nsObj] without using method
Structure Automatic Update Function

This is a function which insures that when an Entity is newly created (ns_New***Data)(※1) or when Entity data is added (ns_Add***Data), consistency is maintained between the structure and Entity data, and that the values of member variables in the created or supplemented contents are updated appropriately.

This function prevents disparity between the structure and the Entity data, and prevents files including inconsistent contents from being created.

Furthermore, for a summary of the member variable for updating each method, and its value, please refer to <Table 2 Summary of Member Variables for Updating Methods>

(※1) When a new entity is created (ns_New***Data), after the structure is initialized, the automatic update function is run.

For initial values, please see <Error! Reference source not found.>

For the initialization object’s structure, please reference each method’s explanation.

Table 2 Summary of Member Variables for Updating Methods

<table>
<thead>
<tr>
<th>Methods Name</th>
<th>Update Object</th>
<th>Structure</th>
<th>Member Variable</th>
<th>Update Variable</th>
<th>Supplementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_NewEventData</td>
<td>ns_FILEINFO</td>
<td>dwEntityCount</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ns_TAGELEMENT</td>
<td>dwElemType</td>
<td>ns_ENTITY_EVENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>dwElemLength</td>
<td>180</td>
<td></td>
<td>Bytes of ns_ENTITYINFO + ns_EVENTINFO</td>
</tr>
<tr>
<td></td>
<td>ns_ENTITYINFO</td>
<td>szEntityLabel</td>
<td>[szEntityLabel statement character string]</td>
<td></td>
<td>New method can’t be changed later</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dwEntityType</td>
<td>ns_ENTITY_EVENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ns_EVENTINFO</td>
<td>dwMinDataLength</td>
<td>$2^{32} - 1$</td>
<td></td>
<td>Obtains correct minimum given the maximum initial value in uint32 form.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dwMaxDataLength</td>
<td>0</td>
<td></td>
<td>Obtains correct maximum given the minimum initial value in uint32 form.</td>
</tr>
<tr>
<td>ns_NewAnalogData</td>
<td>ns_FILEINFO</td>
<td>dwEntityCount</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ns_TAGELEMENT</td>
<td>dwElemType</td>
<td>ns_ENTITY_ANALOG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>dwElemLength</td>
<td>304</td>
<td></td>
<td>Amount of structure ns_ENTITYINFO + ns_ANALOGINFO</td>
</tr>
<tr>
<td></td>
<td>ns_ENTITYINFO</td>
<td>szEntityLabel</td>
<td>[szEntityLabel]</td>
<td></td>
<td>New method can’t be changed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dwEntityType</td>
<td>ns_ENTITY_ANALOG</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### ns_ANALOGINFO

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dMinVal</td>
<td>$2^{32} - 1$</td>
<td>For a given initial value's double form maximum, gives the correct minimum</td>
</tr>
<tr>
<td>dMaxVal</td>
<td>$-2^{32}$</td>
<td>For a given initial value's double form minimum, gives the correct maximum.</td>
</tr>
</tbody>
</table>

### ns_FILEINFO

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwEntityCount</td>
<td>+1</td>
<td></td>
</tr>
</tbody>
</table>

### ns_TAGELEMENT

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwElemType</td>
<td>ns_ENTITY_SEGMENT</td>
<td>Structure ns_ENTITYINFO + ns_SEGMENTINFO amount</td>
</tr>
<tr>
<td>dwElemLength</td>
<td>92</td>
<td></td>
</tr>
</tbody>
</table>

### ns_ENTITYINFO

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>szEntityLabel</td>
<td>[szEntityLabel statement character string]</td>
<td>New method cannot be changed later</td>
</tr>
<tr>
<td>dwEntityType</td>
<td>ns_ENTITY_SEGMENT</td>
<td></td>
</tr>
</tbody>
</table>

### ns_SEGMENTINFO

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwMinSampleCount</td>
<td>$2^{32} - 1$</td>
<td>Obtains correct minimum given the maximum initial value in uint32 form.</td>
</tr>
<tr>
<td>dwMaxSampleCount</td>
<td>0</td>
<td>Obtains correct maximum given the minimum initial value in uint32 form.</td>
</tr>
</tbody>
</table>

### ns_NEWSEGMENTDATA

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwElemType</td>
<td>ns_ENTITY_SEGMENT</td>
<td>Structure ns_ENTITYINFO + ns_SEGMENTINFO amount</td>
</tr>
<tr>
<td>dwElemLength</td>
<td>176</td>
<td></td>
</tr>
</tbody>
</table>

### ns_NEWNEURAL EVENTDATA

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwElemType</td>
<td>ns_ENTITY_NEURAL</td>
<td>Structure ns_ENTITYINFO + ns_NEURALINFO amount</td>
</tr>
<tr>
<td>dwElemLength</td>
<td>+12+</td>
<td>EventValue byte number</td>
</tr>
</tbody>
</table>

### ns_ADDEVENTDATA

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwEventType</td>
<td>[ns_EVENT_***]</td>
<td>Updated depending on data contents</td>
</tr>
<tr>
<td>dwMinDataByteSize</td>
<td>[Min byte size of EventValue]</td>
<td>Updated when the input argument value is less than the minimum until now.</td>
</tr>
<tr>
<td>dwMaxDataByteSize</td>
<td>[Max byte size of EventValue]</td>
<td>Updated when the input argument is greater than the maximum until now.</td>
</tr>
</tbody>
</table>

### ns_ADDANALOGDATA

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwElemLength</td>
<td>+12+8*[analog data quantity]</td>
<td>[...]: Agrees with the length of the vector of the input argument dData</td>
</tr>
</tbody>
</table>

### ns_ADDSEGMENTDATA

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwElemLength</td>
<td>+264+8*[Segment data quantity]</td>
<td>[...]: Agrees with the input argument dValue's vector length</td>
</tr>
<tr>
<td>ns_ENTITYINFO</td>
<td>dwItemCount</td>
<td>+1</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>dwSourceCount</td>
<td>+1</td>
</tr>
<tr>
<td>ns_SEGMENTINFO</td>
<td>dwMinSampleCount</td>
<td>Segment data quantity minimum value</td>
</tr>
<tr>
<td></td>
<td>dwMaxSampleCount</td>
<td>Segment data quantity maximum</td>
</tr>
<tr>
<td>ns_SEGSOURCEINFO</td>
<td>dMinVal</td>
<td>Segment data minimum</td>
</tr>
<tr>
<td></td>
<td>dMaxVal</td>
<td>Segment data maximum</td>
</tr>
<tr>
<td>ns_AddNeuralEventData</td>
<td>dwElemLength</td>
<td>+8</td>
</tr>
<tr>
<td>ns_ENTITYINFO</td>
<td>dwElemCount</td>
<td>+1</td>
</tr>
</tbody>
</table>
Input Value Revision Function[double natural number→uint32]

This is a function which revises double natural number(※1) to uint32 if input needs unsigned integer value.

This function allows user to input even double natural value if the value is saved as uint32. User should not worry about the type of value is double or uint32.

In MATLAB, the default type of num is double. So this function is activated.

(※1)Double natural number: Its type is Double, and its value is equal to integer casted itself. Exam: <double> 5.0. The system treats this as <uint32> 5.

Input Value Revision Function[modify length of character]

This is a function which modifies length of character if input needs char[ x ] value.

This function modifies length of character properly. If strings A’s length was shorter than x, this function fills several blanks. And if it was longer than x, this function cut over length of strings A.

User doesn’t have to care about length of strings.
**ERROR Function**

This is a function which prevents setting wrong value and displays error message on MATLAB Command Window.

For preventing it, system does not create the wrong formatted Neuroshare file.

If user set wrong value, methods which have this function return a ns_Result value below table.

### Table 3 Summary of ns_Result if ERROR occured

<table>
<thead>
<tr>
<th>ns_Result [value]</th>
<th>ERROR message</th>
<th>Reason of ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_FILEERROR 3</td>
<td>FILE MANIPULATION ERROR</td>
<td>File I/O occur.</td>
</tr>
<tr>
<td>ns_WRONGLABEL 101</td>
<td>WRONG DATA_TYPE :LABEL</td>
<td>Wrong type is set.</td>
</tr>
<tr>
<td></td>
<td>WRONG NAME OF OUTPUT_FILE</td>
<td>Wrong output file name is typed.</td>
</tr>
<tr>
<td>ns_WRONGID 102</td>
<td>WRONG ID_TYPE</td>
<td>Wrong id type</td>
</tr>
<tr>
<td></td>
<td>WRONG ID_VALUE</td>
<td>Wrong id value. The entity does not exist.</td>
</tr>
<tr>
<td>ns_WRONGHEADER 103</td>
<td>WRONG INFO :ns_****INFO :This is not correct structure</td>
<td>Wrong structure. The structure does not have enough members.</td>
</tr>
<tr>
<td></td>
<td>WRONG INFO :ns_****INFO :**(The Member doesn't exist)</td>
<td>Wrong structure. The structure has an undefined member.</td>
</tr>
<tr>
<td>ns_WRONGIDATA 104</td>
<td>WRONG DATA_TYPE :<strong><strong>Data :</strong></strong>**</td>
<td>Wrong type is set.</td>
</tr>
</tbody>
</table>

### [Example of the error function]

When a request is made to set an wrong type of value as Entity ID:

```
<< Input to Work Space >> [ ns_Result , nsa_EVENTINFO ] = ns_GetEventInfo( nsObj, 3.2 )
```

### 【 Operation Result 】

Message :  `ERROR : WRONG ID_TYPE : ID (MUST BE 0 or Positive Integer[scalar(1*1)] ) : Stop Sequence.`

ns_Result :  -102 ( ns_WRONGID )

nsObj :  Nothing to be modified.
**WARNING Function**

This is a function which prevents setting wrong value and displays warning message on MATLAB Command Window.

Even warning, system creates the Neuroshare file. But warning members are not updated.

If user set wrong value, methods which have this function return a `ns_Result` value below table.

<table>
<thead>
<tr>
<th><code>ns_Result[Value]</code></th>
<th>WARNING message</th>
<th>Reason of WARNING</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ns_OK</code> 0</td>
<td>WRONG INFO_TYPE : ns_**<strong>INFO.</strong>*</td>
<td>Wrong type is set.</td>
</tr>
<tr>
<td></td>
<td>WRONG INFO_VALUE : ns_**<strong>INFO.</strong>*</td>
<td>Wrong value is set. See Table 5 Summary of region.</td>
</tr>
</tbody>
</table>

[Example of the warning function]

When a request is made to set an wrong type of value as `dwTime_Day`:

```
<<Input to Work Space>>[ ns_Result, nsa_FILEINFO ] = ns_GetFileInfo( nsObj );
<<Input to Work Space>>nsa_FILEINFO.dwTime_Month = 12;
<<Input to Work Space>>nsa_FILEINFO.dwTime_Day = 32;
<<Input to Work Space>>[ ns_Result,nsObj ] = ns_SetFileInfo( nsObj , nsa_FILEINFO );
```

【Operation Result】

Message: WARNING : WRONG INFO_VALUE :ns_FILEINFO.dwTime_Day(MUST BE [1-31]) :This is not updated

ns_Result : 0 ( ns_OK )
nsObj : dwTime_Month is 12, dwTime_Day is not 32.
Method Details

This is a description about methods of the ns_CreateFile class library.

Neuroshare file creation processing

This is a description about methods when you create Neuroshare file.

[Methods]

- ns_CreateFile
  - Create the object to restore Neuroshare data.
  - Create filename, sMagicCode, ns_FILEINFO. Initialize(*1) them.
  - When input argument (filename) is wrong, then returns error value [ns_WRONGLABEL].

- ns_GetFileInfo
  - Get nsa_FILEINFO(*2) data.

- ns_SetFileInfo
  - Update nsa_FILEINFO data.
  - When input argument (nsa_FILEINFO) is wrong, then returns error value [ns_WRONGHEADER], or
    returns ok value [ns_OK] with displaying WARNING MESSAGE.

- ns_CloseFile
  - Integrate all Neuroshare data, Create the Neuroshare file.
  - When file i/o error occurs, then returns error value [ns_FILEERROR].

(*1) See <Table 6 nsObj's contents> about Initial values.
(*2) See <Table 6 nsObj's contents> about members of nsa_***INFO.
[How to use]
You can input command below like (Sample) on MATLAB Work Space.

1. ```
   ns_CreateFile
   (Sample) [retA nsObj] = ns_CreateFile( 'dummy' );
   ```

2. ```
   ns_GetFileInfo
   (Sample) [retB nsa_FILEINFO] = ns_GetFileInfo( nsObj );
   ```

3. Modify members of the structure.
   (Sample) nsa_FILEINFO.szFileComment = 'Words empty of meaning';

4. ```
   ns_SetFileInfo
   (Sample) [retC nsObj] = ns_SetFileInfo( nsObj, nsa_FILEINFO );
   ```

5. Register entities if you need.

6. ```
   ns_CloseFile
   (Sample) [retD] = ns_CloseFile( nsObj );
   ```

Figure 5 ns_CreateFile, ns_CloseFile
Figure 6 ns_GetFileInfo, ns_SetFileInfo
**ns_CreateFile**
Create the object to restore Neuroshare data.

**[How to call]**
[ ns_Result nsObj ] = ns_CreateFile( filename )

**[Summary]**
This method creates the object to restore Neuroshare data, then returns the result data (ns_Result) and the object (nsObj).

**[Arguments]**
Input :
- filename -[char]  the Neuroshare file name(*1).

Output :
- ns_Result -[double] ns_OK or ns_WRONGLABEL.
- nsObj -[struct]  the object to restore Neuroshare data.

[When ns_Result is ns_OK]
the object.

[When ns_Result is not ns_OK]
"" (blank)

**[Note]**
(*1) There are some notes to set filename below.
1. The filename will be the output file name of the Neuroshare.
2. The extension of filename must be 'nsn' or "[nothing].
   If it was wrong, then ERROR occur.
   (exam.) [OK]testfile.nsn, testfile
   [NG]testfile.abc
3. If you wanted to set file arbitrarily, set it with absolute/relative path.
**ns_GetFileInfo**
Get the struct to restore ns_FILEINFO.

**[How to call]**
[ ns_Result nsa_FILEINFO ] = ns_GetFileInfo( nsObj )

**[Summary]**
This method gets the struct to restore ns_FILEINFO, then returns the result data (ns_Result) and the struct (nsa_FILEINFO).

**[Arguments]**
Input :
- nsObj -[struct] the object to restore Neuroshare data.

Output :
- ns_Result -[double] ns_OK.
- nsa_FILEINFO -[struct] the struct to restore nsa_FILEINFO.
**ns_SetFileInfo**
Update the struct to restore ns_FILEINFO.

**[How to call]**
[ ns_Result nsObj ] = ns_SetFileInfo( nsObj, nsa_FILEINFO )

**[Summary]**
This method updates the struct to restore ns_FILEINFO, then returns the result data (ns_Result) and the object (nsObj).

**[Arguments]**
Input :

- **nsObj** - [struct]
  the object to restore Neuroshare data.
- **nsa_FILEINFO** - [struct]
  the struct to restore nsa_FILEINFO(*1).

Output :

- **ns_Result** - [double]
  ns_OK or ns_WRONGHEADER.
- **nsObj** - [struct]
  the object to restore Neuroshare data.

[When ns_Result is ns_OK]
the object which of the ns_FILEINFO is updated.

[When ns_Result is not ns_OK]
""(blank)

**[Note]**

(*1) There are range-limitation of setting members of this struct.

See below table.

### Table 5 Range that can be set

<table>
<thead>
<tr>
<th>Struct name</th>
<th>Member name</th>
<th>Range that can be set</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_FILEINFO</td>
<td>dwTime_Month</td>
<td>[1…12]</td>
<td>January – December</td>
</tr>
<tr>
<td></td>
<td>dwTime_DayOfWeek</td>
<td>[0…6]</td>
<td>0 : Sunday 6 : Saturday</td>
</tr>
<tr>
<td></td>
<td>dwTime_Day</td>
<td>[1…31]</td>
<td>Day</td>
</tr>
<tr>
<td></td>
<td>dwTime_Hour</td>
<td>[0…23]</td>
<td>Hour</td>
</tr>
<tr>
<td></td>
<td>dwTime_Min</td>
<td>[0…59]</td>
<td>Minutes</td>
</tr>
<tr>
<td></td>
<td>dwTime_Sec</td>
<td>[0…59]</td>
<td>Second</td>
</tr>
<tr>
<td></td>
<td>dwTime_MilliSec</td>
<td>[0…1000]</td>
<td>Millisecond</td>
</tr>
</tbody>
</table>
**ns_CloseFile**
Integrate all data.

**[How to call]**
[ ns_Result ] = ns_CloseFile( nsObj )

**[Summary]**
This method integrates the object which restores Neuroshare data and all entities data, creates Neuroshare file(*1). And then, this deletes all intermediate files and returns the result data (ns_Result).

**[Arguments]**
Input :
nsObj - [struct] the object to restore Neuroshare data.

Output :
ns_Result - [double] ns_OK or ns_FILEERROR.

**[Note]**
(*1) The entity's order which the Neuroshare file has obey rule below.
1. Event entities.
2. Analog entities.
3. Segment entities.
4. Neural Event entities.
Event Entity registration processing

This is a description about methods when you register/add event entity.

[Methods]

・ ns_NewEventData
  ➢ Create new event entity to restore Event data.
  ➢ Create ns_TAGELEMENT, ns_ENTITYINFO, ns_EVENTINFO. Initialize(*1) them.
  ➢ Create ID number to identify event entity data.(*2)
  ➢ Create the intermediate file to restore Event data.
  ➢ Update structs(*3).
  ➢ When input argument (szEntityLabel) is wrong, then returns error value [ns_WRONGLABEL].

・ ns_GetEventInfo
  ➢ Get nsa_EVENTINFO(*4) data.
  ➢ When input argument (ID) is wrong, then returns error value [ns_WRONGID].

・ ns_SetEventInfo
  ➢ Update nsa_EVENTINFO data.
  ➢ When input argument (ID) is wrong, then returns error value [ns_WRONGID].
  ➢ When input argument (nsa_EVENTINFO) is wrong, then returns error value [ns_WRONGHEADER],
    or returns ok value [ns_OK] with displaying WARNING MESSAGE.

・ ns_AddEventData
  ➢ Write dTimestamp, dwDataByteSize, EventValue to the intermediate file.
  ➢ Update structs(*3).
  ➢ When input argument (ID) is wrong, then returns error value [ns_WRONGID].
  ➢ When input argument (EventValue) is wrong, then returns error value [ns_WRONGDATA].

(*1) See <Table 6 nsObj's contents> about Initial values.
(*2) Get,Set,Add methods use this ID number to identify event entities.
(*3) See <Table 2 Summary of Member Variables for Updating Methods> about members which are updated.
(*4) See <Table 6 nsObj's contents> about Gettable/Settable member.
[How to use]

You can input command below like (Sample) on MATLAB Work Space.

1. ns_NewEventData
   (Sample) [retD nsObj EventID] = ns_NewEventData( nsObj, 'dummy' );

2. ns_GetEventInfo
   (Sample) [retE nsa_EVENTINFO] = ns_GetEventInfo( nsObj, EventID );

3. Modify members of the structure.
   (Sample) nsa_EVENTINFO.szCSVDesc = 'no meaning string';

4. ns_SetEventInfo
   (Sample) [retF nsObj] = ns_SetEventInfo( nsObj, EventID, nsa_EVENTINFO );

5. ns_AddEventData
   (Sample) [retG nsObj] = ns_AddEventData( nsObj, EventID, 1.5, uint32( 23 ) );

[Notes]
It doesn't correspond to Comma Separated Value. See <Ver1.0>
Figure 7 ns_NewEventData, ns_AddEventData

Figure 8 ns_GetEventInfo, ns_SetEventInfo
**ns_NewEventData**
Create new event entity to restore Event data.

**[How to call]**
[ ns_Result nsObj ID ] = ns_NewEventData( nsObj, szEntityLabel )

**[Summary]**
This method creates the intermediate file to restore Event data, then returns the result data (ns_Result), the object (nsObj) and the identification number (ID).

**[Arguments]**
Input :
- **nsObj** - [struct] the object to restore Neuroshare data.
- **szEntityLabel** - [char] the label of this entity (*1).

Output :
- **ns_Result** - [double] ns_OK or ns_WRONGLABEL.
- **nsObj** - [struct] the object to restore Neuroshare data.
  - When ns_Result is ns_OK
    - the object which is updated (*2).
  - When ns_Result is not ns_OK
    - the object which is NOT updated.
- **ID** - [uint32] the identification number of event data.
  - When ns_Result is ns_OK
    - the identification number (*3).
  - When ns_Result is not ns_OK
    - ""(blank)

**[Note]**
(*1) It is option to input szEntityLabel. If you omitted it, the label will be set as ""(blank).
(*2) See <Table 6 nsObj's contents> about members which are updated.
(*3) It is the order of the Event entity.
ns_GetEventInfo
Get the struct to restore ns_EVENTINFO.

[How to call]
[ ns_Result nsa_EVENTINFO ] = ns_GetEventInfo( nsObj, ID )

[Summary]
This method gets the struct to restore ns_EVENTINFO,
then returns the result data (ns_Result) and the struct (nsa_EVENTINFO).

[Arguments]
Input:

nsObj -[struct] the object to restore Neuroshare data.
ID -[uint32] the identification number of event data.

Output:

ns_Result -[double] ns_OK or ns_WRONGID.
nsa_EVENTINFO-[struct] the struct to restore nsa_EVENTINFO.

[When ns_Result is ns_OK]
the struct which is identified by ID.
[When ns_Result is not ns_OK]
""(blank)
**ns_SetEventInfo**
Update the struct to restore ns_EVENTINFO.

**[How to call]**
[ ns_Result nsObj ] = ns_SetEventInfo( nsObj, ID, nsa_EVENTINFO )

**[Summary]**
This method updates the struct to restore ns_EVENTINFO,
then returns the result data (ns_Result) and the object (nsObj).

**[Arguments]**
Input :
- `nsObj` - [struct] the object to restore Neuroshare data.
- `ID` - [uint32] the identification number of event data.
- `nsa_EVENTINFO` - [struct] the struct to restore nsa_EVENTINFO(*1).

Output :
- `ns_Result` - [double] ns_OK/ns_WRONGID/ns_WRONGHEADER.
- `nsObj` - [struct] the object to restore Neuroshare data.

  [When ns_Result is ns_OK]
  the object which of the ns_EVENTINFO is updated.

  [When ns_Result is not ns_OK]
  the object which is NOT updated.
**ns_AddEventData**

Add new event data to the event entity.

**[How to call]**

\[
\text{[ ns\_Result nsObj ] = ns\_AddEventData( nsObj, ID, dTimestamp, EventValue )}
\]

**[Summary]**

This method adds\(^{(1)}\) event data (dTimestamp and EventValue) to the event entity which is identified by nsObj and ID. Then it returns the result data (ns\_Result) and the object (nsObj).

**[Arguments]**

**Input:**

- **nsObj**
  - [struct] the object to restore Neuroshare data.
- **ID**
  - [uint32] the identification number of event data.
- **dTimestamp**
  - [double] the timestamp value.
- **EventValue**
  - [(u)int8,(u)int16,(u)int32 or char] the event value.\(^{(2)}\)\(^{(3)}\)

**Output:**

- **ns\_Result**
  - [double] ns\_OK/ns\_WRONGID/ns\_WRONGDATA.
- **nsObj**
  - [struct] the object to restore Neuroshare data.
    - [When ns\_Result is ns\_OK]
      - the object which is updated.
    - [When ns\_Result is not ns\_OK]
      - the object which is NOT updated.

**[Note]**

\(^{(1)}\) Add means to write contents below to the intermediate file.

1. dTimestamp
2. dwDataByteSize (It is byte size of the EventValue)
3. EventValue

\(^{(2)}\) It is necessary to set same type of EventValue in the one event ID.

\(^{(3)}\) It doesn't correspond to Comma Separated Value.
Analog Entity registration processing

This is a description about methods when you register/add analog entity.

[Methods]

- **ns_NewAnalogData**
  - Create new analog entity to restore Analog data.
  - Create ns_TAGELEMENT, ns_ENTITYINFO, ns_ANALOGINFO. Initialize\(^{(*)1}\) them.
  - Create ID number to identify analog entity data.\(^{(*)2}\)
  - Create the intermediate file to restore Analog data.
  - Update structs\(^{(*)3}\).
  - When input argument (szEntityLabel) is wrong, then returns error value [ns_WRONGLABEL].

- **ns_GetAnalogInfo**
  - Get nsa_ANALOGINFO\(^{(*)4}\) data.
  - When input argument (ID) is wrong, then returns error value [ns_WRONGID].

- **ns_SetAnalogInfo**
  - Update nsa_ANALOGINFO data.
  - When input argument (ID) is wrong, then returns error value [ns_WRONGID].
  - When input argument (nsa_ANALOGINFO) is wrong, then returns error value [ns_WRONGHEADER], or returns ok value [ns_OK] with displaying WARNING MESSAGE.

- **ns_AddAnalogData**
  - Write dTime, dwDataCount, dData to the intermediate file.
  - Update structs\(^{(*)3}\).
  - When input argument (ID) is wrong, then returns error value [ns_WRONGID].
  - When input argument (dTime, dData) is wrong, then returns error value [ns_WRONGDATA].

\(^{(*)1}\) See <Table 6 nsObj's contents> about Initial values.
\(^{(*)2}\) Get,Set,Add methods use this ID number to identify analog entities.
\(^{(*)3}\) See <Table 2 Summary of Member Variables for Updating Methods> about members which are updated.
\(^{(*)4}\) See <Table 6 nsObj's contents> about Gettable/Settable member.

[How to use]

*You can input command below like (Sample) on MATLAB Work Space.*
1. \texttt{ns\_NewAnalogData}
   
   (Sample) \texttt{[retI nsObj AnalogID] = ns\_NewAnalogData( nsObj, 'dummy' );}

2. \texttt{ns\_GetAnalogInfo}
   
   (Sample) \texttt{[retJ nsa\_ANALOGINFO] = ns\_GetAnalogInfo( nsObj, AnalogID );}

3. Modify members of the structure.
   
   (Sample) \texttt{nsa\_ANALOGINFO.szProbeInfo = 'no meaning string';}

4. \texttt{ns\_SetAnalogInfo}
   
   (Sample) \texttt{[retK nsObj] = ns\_SetAnalogInfo( nsObj, AnalogID, nsa\_ANALOGINFO );}

5. \texttt{ns\_AddAnalogData}
   
   (Sample) \texttt{[retL nsObj] = ns\_AddAnalogData( nsObj, AnalogID, 1.5, [ 5.6 4.5 3.4 ];)

[Notes]

- Nothing.
Figure 9 ns_NewAnalogData, ns_AddAnalogData

Figure 10 ns_GetAnalogInfo, ns_SetAnalogInfo
**ns_NewAnalogData**
Create new analog entity to restore Analog data.

**[How to call]**

```c
[ ns_Result nsObj ID ] = ns_NewAnalogData( nsObj, szEntityLabel )
```

**[Summary]**
This method creates the intermediate file to restore Analog data, then returns the result data (ns_Result), the object (nsObj) and the identification number (ID).

**[Arguments]**

**Input :**

- `nsObj` - [struct] the object to restore Neuroshare data.
- `szEntityLabel` - [char] the label of this entity(*1).

**Output :**

- `ns_Result` - [double] `ns_OK` or `ns_WRONGLABEL`.
- `nsObj` - [struct] the object to restore Neuroshare data.
  - [When ns_Result is ns_OK]
    - the object which is updated(*2).
  - [When ns_Result is not ns_OK]
    - the object which is NOT updated.
- `ID` - [uint32] the identification number of analog data.
  - [When ns_Result is ns_OK]
    - the identification number(*3).
  - [When ns_Result is not ns_OK]
    - “”(blank)

**[Note]**

(*1) It is option to input szEntityLabel. If you omitted it, the label will be set as “”(blank).

(*2) See <Table 6 nsObj’s contents> about members which are updated.

(*3) It is the order of the Analog entity.
**ns_GetAnalogInfo**

Get the struct to restore ns_ANALOGINFO.

[How to call]

[ ns_Result nsa_ANALOGINFO ] = ns_GetAnalogInfo( nsObj, ID )

[Summary]

This method gets the struct to restore ns_ANALOGINFO, then returns the result data (ns_Result) and the struct (nsa_ANALOGINFO).

[Arguments]

Input :

- `nsObj` - [struct] the object to restore Neuroshare data.
- `ID` - [uint32] the identification number of analog data.

Output :

- `ns_Result` - [double] ns_OK or ns_WRONGID.
- `nsa_ANALOGINFO` - [struct] the struct to restore nsa_ANALOGINFO.

  [When ns_Result is ns_OK]
  
  the struct which is identified by ID.

  [When ns_Result is not ns_OK]
  
  "" (blank)
**ns Set Analog Info**

Update the struct to restore ns_ANALOGINFO.

**[How to call]**

\[
\text{[ ns\_Result nsObj ] = ns\_SetAnalogInfo( nsObj, ID, nsa\_ANALOGINFO )}
\]

**[Summary]**

This method updates the struct to restore ns_ANALOGINFO, then returns the result data (ns\_Result) and the object (nsObj).

**[Arguments]**

Input :

- **nsObj** - [struct] the object to restore Neuroshare data.
- **ID** - [uint32] the identification number of analog data.
- **nsa\_ANALOGINFO** - [struct] the struct to restore nsa\_ANALOGINFO(*1).

Output :

- **ns\_Result** - [double] ns\_OK/ns\_WRONGID/ns\_WRONGHEADER.
- **nsObj** - [struct] the object to restore Neuroshare data.

[When ns\_Result is ns\_OK]

the object which of the ns\_ANALOGINFO is updated.

[When ns\_Result is not ns\_OK]

the object which is NOT updated.
**ns_AddAnalogData**

Add new analog data to the analog entity.

**[How to call]**

\[
\text{[ ns\_Result nsObj ] = ns\_AddAnalogData( nsObj, ID, dTime, dData )}
\]

**[Summary]**

This method adds(*1) analog data (dTime and dData) to the analog entity which is identified by nsObj and ID. Then it returns the result data (ns\_Result) and the object (nsObj).

**[Arguments]**

**Input:**

- **nsObj** - [struct] the object to restore Neuroshare data.
- **ID** - [uint32] the identification number of analog data.
- **dTime** - [double] the timestamp value.
- **dData** - [double] the analog value.(vector 1*n)

**Output:**

- **ns\_Result** - [double] ns\_OK/ns\_WRONGID/ns\_WRONGDATA.
- **nsObj** - [struct] the object to restore Neuroshare data.

[When ns\_Result is ns\_OK]

the object which is updated.

[When ns\_Result is not ns\_OK]

the object which is **NOT** updated.

**[Note]**

(*1) Add means to write contents below to the intermediate file.

1. dTime
2. dwDataCount (It is number of the dData)
3. dData (dAnalogValue[0]…dAnalogValue[dwDataCount - 1])
Segment Entity registration processing

This is a description about methods when you register/add segment entity.

[Methods]

• ns_NewSegmentData
  ➢ Create new segment entity to restore Segment data.
  ➢ Create ns_TAGELEMENT, ns_ENTITYINFO, ns_SEGMENTINFO, ns_SEGSOURCEINFO(*5).
    Initialize(*1) them.
  ➢ Create SEGID number to identify segment entity data.(*2)
  ➢ Create the intermediate file to restore Segment data.
  ➢ Update structs(*3).
  ➢ When input argument (szEntityLabel) is wrong, then returns error value [ns_WRONGLABEL].

• ns_GetSegmentInfo
  ➢ Get nsa_SEGMENTINFO(*4) data.
  ➢ When input argument (SEGID) is wrong, then returns error value [ns_WRONGID].

• ns_GetSegmentSourceInfo
  ➢ Get nsa_SEGSOURCEINFO(*4) data.
  ➢ When input argument (SEGID) is wrong, then returns error value [ns_WRONGID].
  ➢ When input argument (SEGSOURCEID) is wrong, then returns error value [ns_WRONGID].

• ns_SetSegmentInfo
  ➢ Update nsa_SEGMENTINFO data.
  ➢ When input argument (SEGID) is wrong, then returns error value [ns_WRONGID].
  ➢ When input argument (nsa_SEGMENTINFO) is wrong, then returns error value
    [ns_WRONGHEADER], or returns ok value [ns_OK] with displaying WARNING MESSAGE.

• ns_SetSegmentSourceInfo
  ➢ Update nsa_SEGSOURCEINFO data.
  ➢ When input argument (SEGID) is wrong, then returns error value [ns_WRONGID].
  ➢ When input argument (SEGSOURCEID) is wrong, then returns error value [ns_WRONGID].
  ➢ When input argument (nsa_SEGSOURCEINFO) is wrong, then returns error value
    [ns_WRONGHEADER], or returns ok value [ns_OK] with displaying WARNING MESSAGE.

• ns_AddSegmentData
- Write dwSampleCount, dTimestamp, dwUnitID, dValue to the intermediate file.
- Create new SEGSOURCEID and add new ns_SEGSOURCEINFO to nsObj. (*5)
- Update structs(*3).
- When input argument (SEGID) is wrong, then returns error value [ns_WRONGID].
- When input argument (dTimestamp, dwUnitID, dValue) is wrong, then returns error value [ns_WRONGDATA].

(*1) See <Table 6 nsObj's contents> about Intial values.
(*2) Get,Set,Add methods use this ID number to identify segment entities/sources.
(*3) See <Table 2 Summary of Member Variables for Updating Methods> about members which are updated.
(*4) See <Table 6 nsObj's contents> about Gettable/Settable member.
(*5) In Ver 1.3[ 1 Segment Entity : 1 SegSourceInfo Header.], ns_SEGSOURCEINFO is unique and created by ns_NewSegmentData(). (Not ns_AddSegmentData())

[How to use]
You can input command below like (Sample) on MATLAB Work Space.
1. ns_NewSegmentData
   (Sample) [retM nsObj SEGID] = ns_NewSegmentData( nsObj, 'dummy' );

2. ns_GetSegmentInfo
   (Sample) [retN nsa_SEGMENTINFO] = ns_GetSegmentInfo( nsObj, SegmentID );

3. Modify members of the structure.
   (Sample) nsa_SEGMENTINFO.szUnits = 'no meaning';

4. ns_SetSegmentInfo
   (Sample) [retO nsObj] = ns_SetSegmentInfo( nsObj, SEGID, nsa_SEGMENTINFO );

5. ns_GetSegmentSourceInfo
   (Sample) [retQ nsa_SEGSOURCEINFO] = ns_GetSegmentSourceInfo( nsObj, SEGID, SEGSOURCEID );

6. Modify members of the structure.
   (Sample) nsa_SEGSOURCEINFO.szProbeInfo = 'no meaning string';

7. ns_SetSegmentInfo
   (Sample) [retR nsObj] = ns_SetSegmentSourceInfo( nsObj, SEGID, SEGSOURCEINFO,
8. ns_AddSegmentData

(Sample) [retP nsObj SEGSOURCEID] = ns_AddSegmentData( nsObj, SEGID, 1.5, 3, [ 5.6 4.5 3.4 ] );

[Notes]
It is impossible to call methods ns_GetSegmentSourceinfo and ns_SetSegmentSourceInfo before call the method ns_AddSegmentData().
Because SEGSOURCEID and ns_SEGSOURCEINFO are created when you call the method ns_AddSegmentData().
ns_SEGSOURCEINFO is unique. So, if you call ns_GetSegmentSourceInfo(nsObj, SEGID, 2), ERROR will be occurred.
Figure 11 ns_NewSegmentData, ns_AddSegmentData

Figure 12 ns_GetSegmentInfo, ns_SetSegmentInfo

Figure 13 ns_GetSegmentSourceInfo, ns_SetSegmentSourceInfo
**ns NewSegmentData**

Create new segment entity to restore Segment data.

[How to call]

[ ns_Result nsObj SEGID ] = ns_NewSegmentData( nsObj, szEntityLabel )

[Summary]

This method creates the intermediate file to restore Segment data and ns_SEGSOURCEINFO(*4), then returns the result data (ns_Result), the object (nsObj) and the identification number (SEGID).

[Arguments]

Input :

- **nsObj** - [struct] the object to restore Neuroshare data.
- **szEntityLabel** - [char] the label of this entity(*1).

Output :

- **ns_Result** - [double] ns_OK or ns_WRONGLABEL.
- **nsObj** - [struct] the object to restore Neuroshare data.
  - [When ns_Result is ns_OK] the object which is updated(*2).
  - [When ns_Result is not ns_OK] the object which is NOT updated.
- **SEGID** - [uint32] the identification number of analog data.
  - [When ns_Result is ns_OK] the identification number(*3).
  - [When ns_Result is not ns_OK] ""(blank)

[Note]

(*1) It is option to input szEntityLabel. If you omitted it, the label will be set as ""(blank).
(*2) Refer Table 6 about members which are updated.
(*3) It is the order of the Segment entity.
(*4) From Ver 1.3
**ns_GetSegmentInfo**

Get the struct to restore ns_SEGMENTINFO.

**[How to call]**

\[
\text{[ ns\_Result nsa\_SEGMENTINFO ] = ns\_GetSegmentInfo( nsObj, SEGID )}
\]

**[Summary]**

This method gets the struct to restore ns_SEGMENTINFO, then returns the result data (ns\_Result) and the struct (nsa\_SEGMENTINFO).

**[Arguments]**

**Input:**

- \( \text{nsObj} \) - [struct] the object to restore Neuroshare data.
- \( \text{SEGID} \) - [uint32] the identification number of segment data.

**Output:**

- \( \text{ns\_Result} \) - [double] ns\_OK or ns\_WRONGID.
- \( \text{nsa\_SEGMENTINFO} \) - [struct] the struct to restore nsa\_SEGMENTINFO.
  - [When \( \text{ns\_Result} \) is ns\_OK]
    - the struct which is identified by SEGID.
  - [When \( \text{ns\_Result} \) is not ns\_OK]
    - "" (blank)
**ns_GetSegmentSourceInfo**
Get the struct to restore ns_SEGSOURCEINFO.

[How to call]
[ ns_Result nsa_SEGSOURCEINFO ] = ns_GetSegmentSourceInfo( nsObj, SEGID, SEGSOURCEID )

[Summary]
This method gets the struct to restore ns_SEGSOURCEINFO,
then returns the result data (ns_Result) and the struct (nsa_SEGMENTINFO).

[Arguments]
Input :
nsObj    -[struct]    the object to restore Neuroshare data.
SEGID    -[uint32]    the identification number of segment data.
SEGSOURCEID  -[uint32]  the identification number of segment source.

Output :
ns_Result    -[double]    ns_OK or ns_WRONGID.
nsa_SEGSOURCEINFO-[struct]    the struct to restore nsa_SEGSOURCEINFO.
    [When ns_Result is ns_OK]
        the struct which is identified by SEGID and SEGSOURCEID.
    [When ns_Result is not ns_OK]
        "" (blank)
ns_SetSegmentInfo
Update the struct to restore ns_SEGMENTINFO.

[How to call]
[ ns_Result nsObj ] = ns_SetSegmentInfo( nsObj, SEGID, nsa_SEGMENTINFO )

[Summary]
This method updates the struct to restore ns_SEGMENTINFO, then returns the result data (ns_Result) and the object (nsObj).

[Arguments]
Input :

- nsObj - [struct] the object to restore Neuroshare data.
- SEGID - [uint32] the identification number of segment data.
- nsa_SEGMENTINFO - [struct] the struct to restore nsa_SEGMENTINFO(*1).

Output :

- ns_Result - [double] ns_OK/ns_WRONGID/ns_WRONGHEADER.
- nsObj - [struct] the object to restore Neuroshare data.

[When ns_Result is ns_OK]
The object which of the ns_SEGMENTINFO is updated.

[When ns_Result is not ns_OK]
The object which is NOT updated.
**ns_SetSegmentSourceInfo**
Update the struct to restore ns_SEGSOURCEINFO.

**[How to call]**
[ ns_Result nsObj ] = ns_SetSegmentSourceInfo( nsObj, SEGID, SEGSOURCEID, nsa_SEGSOURCEINFO )

**[Summary]**
This method updates the struct to restore ns_SEGSOURCEINFO, then returns the result data (ns_Result) and the object (nsObj).

**[Arguments]**
Input:
- **nsObj** - [struct] the object to restore Neuroshare data.
- **SEGID** - [uint32] the identification number of segment data.
- **SEGSOURCEID** - [uint32] the identification number of segment source.
- **nsa_SEGSOURCEINFO** - [struct] the struct to restore nsa_SEGSOURCEINFO(*1).

Output:
- **ns_Result** - [double] ns_OK/ns_WRONGID/ns_WRONGHEADER.
- **nsObj** - [struct] the object to restore Neuroshare data.

[When ns_Result is ns_OK]
the object which of the ns_SEGSOURCEINFO is updated.

[When ns_Result is not ns_OK]
the object which is NOT updated.
**ns_AddSegmentData**
Add new segment data to the segment entity.

**[How to call]**

\[ \text{[ns\_Result ns\_Obj SEGSOURCEID ] = ns\_AddSegmentData( ns\_Obj, SEGID, dTimestamp, dwUnitID, dValue) \} \] (*3)

\[ \text{[ ns\_Result ns\_Obj ] = ns\_AddSegmentData( ns\_Obj, SEGID, dTimestamp, dwUnitID, dValue )} \]

**[Summary]**
This method adds new `ns_SEGSOURCEINFO` (identified by SEGSOURCEID) (*1) (*4) segment data (dTimestamp, dwUnitId and dValue) (*2) to the segment entity which is identified by `ns_obj` and SEGID. Then it returns the result data (ns_Result), the object (nsObj) and the new identified number(SEGSOURCEID).

**[Arguments]**

**Input :**
- `nsObj` - [struct] the object to restore Neuroshare data.
- `SEGID` - [uint32] the identification number of segment data.
- `dTimestamp` - [double] the timestamp value.
- `dUnitID` - [uint32] the Classification ID.
- `dValue` - [double] the segment value.(vector 1*n)

**Output :**
- `ns\_Result` - [double] ns\_OK/ns\_WRONGID/ns\_WRONGDATA.
- `nsObj` - [struct] the object to restore Neuroshare data.

  [When ns\_Result is ns\_OK]
  the object which is updated.
  [When ns\_Result is not ns\_OK]
  the object which is NOT updated.

  **SEGSOURCEID** - [uint32] the identification number of segment source.

  [When ns\_Result is ns\_OK]
  the identification number(*3).
  [When ns\_Result is not ns\_OK]
  = (blank) (*3)

**[Note]**

(*1) It causes the consistency between the record number of this entity and the number of ns_SEGSOURCEINFO that creating new ns_SEGSOURCEINFO when adding a segment data.

(*2) Add means to write contents below to the intermediate file.
1. dwSampleCount (It is number of the dValue)
2. dTimestamp
3. dwUnitID
4. dValue (dValue[0]…dAnalogValue[dwSampleCount - 1])

(*3) I/F was modified.[Ver. 1.3]
(*4) It moved to the ns_NewSegmentData().[Ver. 1.3]
Neural Event Entity registration processing

This is a description about methods when you register/add neural event entity.

[Methods]

- **ns_NewNeuralEventData**
  - Create new neural event entity to restore Neural Event data.
  - Create ns_TAGELEMENT, ns_ENTITYINFO, ns_NEURALINFO. Initialize(*1) them.
  - Create ID number to identify neural event entity data.(*2)
  - Create the intermediate file to restore Neural Event data.
  - Update structs(*3).
  - When input argument (szEntityLabel) is wrong, then returns error value [ns_WRONGLABEL].

- **ns_GetNeuralInfo**
  - Get nsa_NEURALINFO(*4) data.
  - When input argument (ID) is wrong, then returns error value [ns_WRONGID].

- **ns_SetNeuralInfo**
  - Update nsa_NEURALINFO data.
  - When input argument (ID) is wrong, then returns error value [ns_WRONGID].
  - When input argument (nsa_NEURALINFO) is wrong, then returns error value [ns_WRONGHEADER],
    or returns ok value [ns_OK] with displaying WARNING MESSAGE.

- **ns_AddNeuralEventData**
  - Write dTimestamp to the intermediate file.
  - Update structs(*3).
  - When input argument (ID) is wrong, then returns error value [ns_WRONGID].
  - When input argument (dTimestamp) is wrong, then returns error value [ns_WRONGDATA].

(*1) See <Table 6 nsObj's contents> about Intial values.
(*2) Get,Set,Add methods use this ID number to identify neural event entities.
(*3) See <Table 2 Summary of Member Variables for Updating Methods> about members which are updated.
(*4) See <Table 6 nsObj's contents> about Gettable/Settable member.

[How to use]

You can input command below like (Sample) on MATLAB Work Space.
1. ns_NewNeuralEventData  
   (Sample) [retS nsObj NeuralID] = ns_NewNeuralEventData( nsObj, 'dummy' );

2. ns_GetNeuralInfo  
   (Sample) [retT nsa_NEURALINFO] = ns_GetNeuralInfo( nsObj, NeuralID );

3. Modify members of the structure.  
   (Sample) nsa_NEURALINFO.szProbeInfo = 'no meaning string';

4. ns_SetNeuralInfo  
   (Sample) [retU nsObj] = ns_SetNeuralInfo( nsObj, NeuralID, nsa_NEURALINFO );

5. ns_AddNeuralEventData  
   (Sample) [retV nsObj] = ns_AddNeuralEventData( nsObj, NeuralID, 1.5 );

[Notes]  
・ You have to know how many entities are included in the Neuroshare file if you wanted to set  
   ns_NEURALINFO correctly. See <Ver 1.0>.
Figure 14 ns_NewNeuralEventData, ns_AddNeuralEventData

Figure 15 ns_GetNeuralInfo, ns_SetNeuralInfo
**ns_NewNeuralEventData**
Create new neural event entity to restore Neural Event data.

**[How to call]**

[ ns_Result nsObj ID ] = ns_NewNeuralEventData( nsObj, szEntityLabel )

**[Summary]**
This method creates the intermediate file to restore Neural Event data, then returns the result data (ns_Result), the object (nsObj) and the identification number (ID).

**[Arguments]**

**Input:**
- **nsObj** - [struct] the object to restore Neuroshare data.
- **szEntityLabel** - [char] the label of this entity(*1).

**Output:**
- **ns_Result** - [double] ns_OK or ns_WRONGLABEL.
- **nsObj** - [struct] the object to restore Neuroshare data.
  - [When ns_Result is ns_OK] the object which is updated(*2).
  - [When ns_Result is not ns_OK] the object which is NOT updated.
- **ID** - [uint32] the identification number of neural event data.
  - [When ns_Result is ns_OK] the identification number(*3).
  - [When ns_Result is not ns_OK] “”(blank)

**[Note]**

(*1) It is option to input szEntityLabel. If you omitted it, the label will be set as “”(blank).
(*2) See <Table 6 nsObj's contents> about members which are updated.
(*3) It is the order of the Neural Event entity.
**ns_GetNeuralInfo**
Get the struct to restore ns_NEURALINFO.

**[How to call]**
[ ns_Result nsa_NEURALINFO ] = ns_GetNeuralInfo( nsObj, ID )

**[Summary]**
This method gets the struct to restore ns_NEURALINFO, then returns the result data (ns_Result) and the struct (nsa_NEURALINFO).

**[Arguments]**
Input :
- nsObj -[struct] the object to restore Neuroshare data.
- ID -[uint32] the identification number of neural event data.

Output :
- ns_Result -[double] ns_OK or ns_WRONGID.
- nsa_NEURALINFO-[struct] the struct to restore nsa_NEURALINFO.

[When ns_Result is ns_OK]
- the struct which is identified by ID.
[When ns_Result is not ns_OK]
- ""(blank)
**ns_SetNeuralInfo**

Update the struct to restore ns_NEURALINFO.

**[How to call]**

\[
\text{[ ns\_Result nsObj ] = ns\_SetNeuralInfo( nsObj, ID, nsa\_NEURALINFO )}
\]

**[Summary]**

This method updates the struct to restore ns_NEURALINFO, then returns the result data (ns\_Result) and the object (nsObj).

**[Arguments]**

**Input:**

- `nsObj` - [struct] the object to restore Neuroshare data.
- `ID` - [uint32] the identification number of neural event data.
- `nsa\_NEURALINFO` - [struct] the struct to restore nsa\_NEURALINFO(*1).

**Output:**

- `ns\_Result` - [double] ns\_OK/ns\_WRONGID/ns\_WRONGHEADER.
- `nsObj` - [struct] the object to restore Neuroshare data.

[When ns\_Result is ns\_OK]

- the object which of the ns\_NEURALINFO is updated.

[When ns\_Result is not ns\_OK]

- the object which is NOT updated.
**ns_AddNeuralEvent**

Add new neural event data to the neutral event entity.

**[How to call]**

\[ \text{ns\_Result nsObj} ] = \text{ns\_AddNeuralEvent}( \text{nsObj, ID, dTime} )

**[Summary]**

This method adds\(^{(1)}\) neural event data (\(dTime\)) to the neural event entity which is identified by \(\text{nsObj}\) and \(\text{ID}\). Then it returns the result data (\(\text{ns\_Result}\)) and the object (\(\text{nsObj}\)).

**[Arguments]**

**Input:**

- **nsObj** - [struct] the object to restore Neuroshare data.
- **ID** - [uint32] the identification number of neural event data.
- **dTime** - [double] the timestamp value.

**Output:**

- **ns\_Result** - [double] \(\text{ns\_OK/ns\_WRONGID/ns\_WRONGDATA}\).
- **nsObj** - [struct] the object to restore Neuroshare data.

[When \(\text{ns\_Result}\) is \(\text{ns\_OK}\)]

the object which is updated.

[When \(\text{ns\_Result}\) is not \(\text{ns\_OK}\)]

the object which is NOT updated.

**[Note]**

\(^{(1)}\) Add means to write contents below to the intermediate file.

1. \(dTime\)
**Terminology Supplement**

### nsObj

nsObj is the object of ns_CreateFile class library and stores the necessary information to create the Neuroshare file. See <Figure 16 Format of nsObj> and <Table 6 nsObj's contents>.

### nsa_***INFO

nsa_***INFO is the struct which includes some members of ns_***INFO. nsa_***INFO only has modifiable members from ns_***INFO. See <Table 6 nsObj's contents>.

You can only modify nsa_***INFO members. The reason of it is to avoid unconsistency with header data and real data. For example, in ns_FILEINFO, the member dwEntityCount is unmodifiable. The meaning of dwEntityCount indicates the number of entities which the Neuroshare file has. If the value was modifiable, it causes inconsistency between ns_FILEINFO (i.e. header data) and real data.
Figure 16 Format of nsObj
<table>
<thead>
<tr>
<th>Struct name</th>
<th>Member name</th>
<th>Type</th>
<th>Default Value</th>
<th>Comment</th>
<th>Modifiable of not with using Getter/Setter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sMagicCode</td>
<td>char[16]</td>
<td>&quot;NSN ver00000000010&quot;</td>
<td>Document type identifier</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>filename</td>
<td>char</td>
<td>Nothing</td>
<td>The Neuroshare file name. You have to set it when use class.</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>szFileType</td>
<td>char[32]</td>
<td>&quot;(blank)&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ns_FILEINFO</td>
<td>dwEntityCount</td>
<td>uint32</td>
<td>0</td>
<td>Entity number.</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>dTimeStampResolution</td>
<td>double</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dTimeSpan</td>
<td>double</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>szAppName</td>
<td>char[64]</td>
<td>&quot;(blank)&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwTime_Year</td>
<td>uint32</td>
<td>1900</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwTime_Month</td>
<td>uint32</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwTime_DayOFWeek</td>
<td>uint32</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwTime_Day</td>
<td>uint32</td>
<td>1</td>
<td>Treat Jan/1/1900 Mon 00:00:00:0000 as default value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwTime_Hour</td>
<td>uint32</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwTime_Min</td>
<td>uint32</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwTime_Sec</td>
<td>uint32</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwTime_MilliSec</td>
<td>uint32</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>szFileComment</td>
<td>char[256]</td>
<td>&quot;(blank)&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwElemType</td>
<td>uint32</td>
<td>nsObj.CONST.ns_ENTITY_UNKNOWN</td>
<td>Type of entity. Default is unknown.</td>
<td>×</td>
</tr>
<tr>
<td>ns_TAGELEMENT</td>
<td>dwElemLength</td>
<td>uint32</td>
<td>0</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>szEntityLabel</td>
<td>char[32]</td>
<td>&quot;(blank)&quot;</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>ns_ENTITYINFO</td>
<td>dwEntityType</td>
<td>uint32</td>
<td>nsObj.CONST.ns_ENTITYUNKNOWN</td>
<td>Type of entity. Default is unknown.</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>dwItemCount</td>
<td>uint32</td>
<td>0</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>dwEventType</td>
<td>uint32</td>
<td>nsObj.CONST.ns_EVENT_TEXT</td>
<td>Type of Event Data. Default is Text.</td>
<td>×</td>
</tr>
<tr>
<td>ns_EVENTINFO</td>
<td>dwMinDataLength</td>
<td>uint32</td>
<td>0</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>dwMaxDataLength</td>
<td>uint32</td>
<td>0</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>szCSVDesc</td>
<td>char[128]</td>
<td>&quot;(blank)&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dSampleRate</td>
<td>double</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dMinVal</td>
<td>double</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dMaxVal</td>
<td>double</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>szUnits</td>
<td>char[16]</td>
<td>&quot;(blank)&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ns_ANALOGINFO</td>
<td>dResolution</td>
<td>double</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dLocationX</td>
<td>double</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dLocationY</td>
<td>double</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ns_SEGMENTINFO</td>
<td>dwSourceCount</td>
<td>uint32</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------</td>
<td>---------</td>
<td>---------</td>
<td>-----------</td>
<td>----</td>
</tr>
<tr>
<td></td>
<td>dwMinSampleCount</td>
<td>uint32</td>
<td>0</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dwMaxSampleCount</td>
<td>uint32</td>
<td>0</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dSampleRate</td>
<td>double</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>szUnits</td>
<td>char[32]</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

| ns_SEGSOURCEINFO       | dMinVal           | double  | 0       |            | 0  |
|                        | dMaxVal           | double  | 0       |            | 0  |
|                        | dResolution       | double  | 0       |            | 0  |
|                        | dSubSampleShift   | double  | 0       |            | 0  |
|                        | dLocationX        | double  | 0       |            | 0  |
|                        | dLocationY        | double  | 0       |            | 0  |
|                        | dLocationZ        | double  | 0       |            | 0  |
|                        | dLocationUser     | double  | 0       |            | 0  |
|                        | dhHighFreqCorner  | double  | 0       |            | 0  |
|                        | dwHighFreqOrder   | uint32  | 0       |            | 0  |
|                        | szHighFilterType  | char[16]| "(blank)|            | 0  |
|                        | dLowFreqCorner    | double  | 0       |            | 0  |
|                        | dwLowFreqOrder    | uint32  | 0       |            | 0  |
|                        | szLowFilterType   | char[16]| "(blank)|            | 0  |
|                        | szProbeInfo       | char[128]| "(blank)|            | 0  |

| ns_NEURALINFO          | dwSourceEntityID  | uint32  | 0       |            | 0  |
|                        | dwSourceUnitID    | uint32  | 0       |            | 0  |
|                        | szProbeInfo       | char[128]| "(blank)|            | 0  |
Notice Points

Ver 1.0

- About ns_NEURALINFO.dwSourceEntityID and ns_NEURALINFO.dwSourceUnitID.
  If you wanted to set these members correctly, you have to realize how many entities exist in the Neuroshare file. The reason of it is that this library creates the Neuroshare file with the order (Event, Analog, Segment, NeuralEvent).

- About CSV data format.
  The method ns_AddEventData() doesn't correspond to the input of Comma Separated Value.

Ver 1.3

- 1 Segment Entity : 1 SegSourceInfo Header.
  In Ver 1.3, we defined the relationship num of SegSourceInfo and SegmentData as 1:N.
  So, you can register "ONE" SegSourceInfo header per one Segment entity.

  Old:
  Num of Segment Entity : Num of SegSourceInfo : Num of SegmentData records = 1:N:N

  New:
  Num of Segment Entity : Num of SegSourceInfo : Num of SegmentData records = 1:1:N

  Then, two methods were modified. ns_NewSegmentData() and ns_AddSegmentData().
  See docs if you want to know more.
Update Log

Ver 1.0 Date : 2009/06/25
New Creation with referring site below.
POMU-LAB site : http://www2.bpe.es.osaka-u.ac.jp/multineuron/POMU-Lab/index.html

Ver 1.3 Date : 2011/03/04
Modified for Ver 1.3.

Acknowledgment

Special thanks for translating this document to T. Beck.