ns_CreateFile クラスライブラリ解説資料

Ver. 1.3

2011/03/04

はじめに

本資料は、ns_CreateFile クラスライブラリの解説資料です。

Neuroshare 形式(※1)の構成を把握した上で本資料を読むことをお薦めします。

ns_CreateFile クラスライブラリは、標準的な神経生理学実験のデータを、矛盾のない Neuroshare 形式のデータに変換する関数(以降メソッド)群です。

ns_CreateFile クラスライブラリの動作確認は、Windows XP SP2 ,MATLAB 7.5.0 (R2007b)の環境で行っております。

尚、ns_CreateFile クラスライブラリを使用、かつダミーのデータを用いたデータ変換の例は、

アプリケーション ：SampleConverter.m において実装されています。

ns_CreateFile クラスライブラリの具体的な使用例についてはそちらをご覧下さい。

(※1)　Neuroshare 形式 ： 神経生理学実験データの共通フォーマット。拡張子は(.nsn)。
4 つの Entity 種別の定義と、構造体と Entity データの組み合わせによって、
神経生理学実験データを表現する。
<図 1 Neuroshare Format>と Neuroshare Native File Format Specification Rev 0.9d を
参照してください。 Neuroshare site : http://neuroshare.sourceforge.net/download.shtml

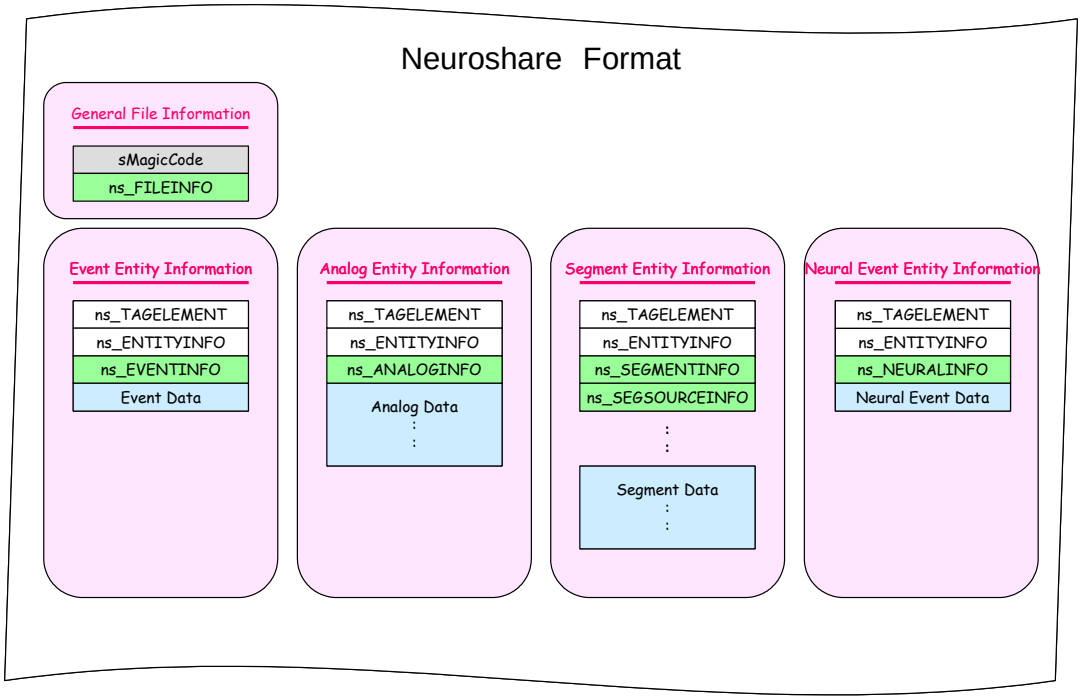


図 1 Neuroshare Format

ns_CreateFile クラスライブラリの目的

ns_CreateFile クラスライブラリの目的について記載しています。

ns_CreateFile クラスライブラリの目的は、
「標準的な神経生理学実験のデータ(以降実験データ)を元に、正しい Neuroshare 形式のデータに変換する(※1)こと」です。

また、ns_CreateFile クラスライブラリが正しい Neuroshare 形式のデータを作成するよう、
そのメソッドは適切な機能を持ちます。

本解説資料では、ns_CreateFile クラスライブラリのメソッド一覧、使用する定数、Neuroshare ファイル作成の流れ、複数のメソッドに共通した機能、個々のメソッドに特化した機能の順で、次項よりそれぞれ解説しています。

(※1)「正しい Neuroshare 形式のデータに変換する」とは、
ns_CreateFile クラスライブラリによって実験データを元に作成したファイルが、下記条件を満たすことです。

1. ファイル中全ての構造体に記載されている内容が、Neuroshare 形式における定義と一致している
2. ファイル中全ての Entity データに記載されている内容が、Neuroshare 形式における定義と一致している
3. ファイル中の構造体と Entity データの間で記載される内容に差異が生じていない

メソッド一覧

ns_CreateFile クラスライブラリ内には、以下のメソッドが存在します。

メソッド名と、その機能要約を記載しています。各メソッドの詳細な機能については<メソッド詳細>を参照してください。

[メソッド名]	[機能要約]
<Neuroshare 形式ファイル作成処理>	
ns_CreateFile	Neuroshare データ格納用オブジェクト(以降 nsObj) (※1)の作成を行う
ns_GetFileInfo	ファイル情報構造体(以降 nsa_FILEINFO(※2))の取得を行う
ns_SetFileInfo	nsa_FILEINFO(※2)の更新を行う
ns_CloseFile	全てのデータの統合を行う
<Event Entity 登録処理>	
ns_NewEventData	新規イベントデータの生成処理を行う
ns_GetEventInfo	イベント情報構造体(以降 nsa_EVENTINFO(※2))の取得を行う
ns_SetEventInfo	nsa_EVENTINFO(※2)の更新を行う
ns_AddEventData	イベントデータの追加を行う
<Analog Entity 登録処理>	
ns_NewAnalogData	新規アナログデータの生成処理を行う
ns_GetAnalogInfo	アナログ情報構造体(以降 nsa_ANALOGINFO(※2))の取得を行う
ns_SetAnalogInfo	nsa_ANALOGINFO(※2)の更新を行う
ns_AddAnalogData	アナログデータの追加を行う
<Segment Entity 登録処理>	
ns_NewSegmentData	新規セグメントデータの生成処理を行う
ns_GetSegmentInfo	セグメント情報構造体(以降 nsa_SEGMENTINFO(※2))の取得を行う
ns_GetSegmentSourceInfo	セグメントソース情報構造体(以降 nsa_SEGSOURCEINFO(※2))の取得を行う
ns_SetSegmentInfo	nsa_SEGMENTINFO(※2)の更新を行う
ns_SetSegmentSourceInfo	nsa_SEGSOURCEINFO(※2)の更新を行う
ns_AddSegmentData	ns_SEGSOURCEINFO の追加と、セグメントデータの追加を行う
<Neural Event Entity 登録処理>	
ns_NewNeuralEventData	新規ニューラルイベントデータの生成処理を行う

ns_GetNeuralInfo	ニューラルイベント情報構造体(以降 nsa_NEURALINFO（※2）)の取得を行う
ns_SetNeuralInfo	nsa_NEURALINFO（※2）の更新を行う
ns_AddNeuralEventData	ニューラルイベントデータの追加を行う

<その他機能>

display	構造体-メンバ変数の現在値を表示する(ユーザーによる設定可能な箇所のみを表示)
display_All_Infomation	各構造体-メンバ変数の現在値を表示する(全メンバ変数を表示)
subsasgn	.(ドット)演算子を利用した各構造体-メンバ変数の更新を禁止する

（※1） nsObj は、「Neuroshare 形式のデータを格納するオブジェクト」です。
詳しくは<nsObj とは>を参照してください。

（※2） nsa_***INFO 構造体は、「ns_***INFO 構造体から
Entity データとの整合性に関わらないメンバ変数を抽出した構造体」のことです。
詳しくは<nsa_***INFO 構造体とは>を参照してください。

使用定数一覧

ns_CreateFile クラスライブラリ内で使用している定数値の一覧を示します。

ns_CreateFile クラスライブラリで使用する定数値は、下記<表 1 使用定数一覧>の通りです。
Neuroshare Matlab API Specification Rev2.2(※1)で定義されている定数に、
ns_CreateFile クラスライブラリで使用する新たな定数を追加したものとなっています。

表 1 使用定数一覧

定数名	実際の値	定数の意味
< メソッド戻り値 >※背景色橙色の箇所は ns_CreateFile クラスライブラリにおいて新規に作成した定数		
ns_OK	0	メソッド処理正常終了 (WARNING 処理時も含む。<WARNING 処理機能について>を参照)
ns_LIBERROR	-1	ライブラリリンクエラー発生
ns_TYPEERROR	-2	ファイルオープンエラー発生
ns_FILEERROR	-3	ファイルアクセスエラー発生
ns_BADFILE	-4	ファイルハンドルエラー発生
ns_BADENTITY	-5	EntityID 記述エラー発生
ns_BADSOURCE	-6	SourceD 記述エラー発生
ns_BADINDEX	-7	インデックスエラー発生
ns_WRONGLABEL	-101	入力引数の文字列の指定にエラー (ファイル名の指定もしくは EntityLabel の指定エラー)
ns_WRONGID	-102	入力引数の EntityID もしくは SegmentSourceID の指定にエラー
ns_WRONGHEADER	-103	入力引数の構造体の内容にエラー
ns_WRONGDATA	-104	入力引数のデータの内容にエラー
< メンバ変数への設定値 : ns_TAGELEMENT ns_ENTITYINFO >		
ns_ENTITY_UNKOWN	0	Entity タイプ不明
ns_ENTITY_EVENT	1	イベント Entity
ns_ENTITY_ANALOG	2	アナログ Entity
ns_ENTITY_SEGMENT	3	セグメント Entity
ns_ENTITY_UNKOWN	4	ニューラルイベント Entity
ns_INFO_FILE	5	ファイル情報
< メンバ変数への設定値 : ns_EVENTINFO >		

ns_EVENT_TEXT	0	文字列
ns_EVENT_CSV	1	コンマ区切り値
ns_EVENT_BYTE	2	8 ビットバイナリ地
ns_EVENT_WORD	3	16 ビットバイナリ値
ns_EVENT_DWORD	4	32 ビットバイナリ値

(※1)Neuroshare Matlab API Specification Rev2.2 : Neuroshare 形式のファイルを読み込むライブラリ。

Neuroshare Matlab API Specification Rev2.2 を参照してください。

Neuroshare site : http://neuroshare.sourceforge.net/download.shtml

Neuroshare 形式ファイル作成の流れ

Neuroshare 形式のファイルを作成するまでの一連の流れは下記の通りです。

MATLAB – Work Space 上でのユーザー作業を含め、メソッドを使用順に並べると、
下記 1.–10.の通りになります。(下線付きがメソッド名)

また、下記 1.–10.を行うと、nsObj に対して各種変更が行われます。
<図 2 SampleData 作成フロー[1]>から<図 4 SampleData 作成フロー[3]>を参照してください。
尚、イベントデータを登録する場合を例にとっています。(>> は Work Space 上での入力例)

1. ns_CreateFile ・・・Neuroshare 形式ファイルの作成を行う
   >> [retA nsObj] = ns_CreateFile('sample.nsn');
2. ns_GetFileInfo ・・・nsa_FILEINFO の取得を行う
   >> [retB nsa_FILEINFO] = ns_GetFileInfo(nsObj);
3. [ユーザーによる構造体の内容の変更]
   >> nsa_FILEINFO.szFileComment = 'Sample';
4. ns_SetFileInfo ・・・nsa_FILEINFO の更新を行う
   >> [retC nsObj] = ns_SetFileInfo(nsObj, nsa_FILEINFO);
5. ns_NewEventData ・・・新規イベントデータの生成処理を行う
   >> [retD nsObj EventID] = ns_NewEventData(nsObj, 'dummy');
6. ns_GetEventInfo ・・・nsa_EVENTINFO の取得を行う
   >> [retE nsa_EVENTINFO] = ns_GetEventInfo(nsObj, EventID);
7. [ユーザーによる構造体の内容の変更]
   >> nsa_EVENTINFO.szCSVDesc = 'Words empty of meaning';
8. ns_SetEventInfo ・・・nsa_EVENTINFO の更新を行う
   >> [retF nsObj] = ns_SetEventInfo(nsObj, EventID, nsa_EVENTINFO);
9. ns_AddEventData ・・・イベントデータの追加を行う
   >> [retG nsObj] = ns_AddEventData(nsObj, EventID, 1.5, uint32(23));
10. ns_CloseFile ・・・全てのデータの統合
   >> [retH] = ns_CloseFile(nsObj);

※同一の EntityID にイベントデータを複数登録する場合は、9.を繰り返し使用します。
※複数の EntityID を登録する場合は、再度 5.–9.を使用します。
※全てのデータを登録し終えたならば、10.を使用します。

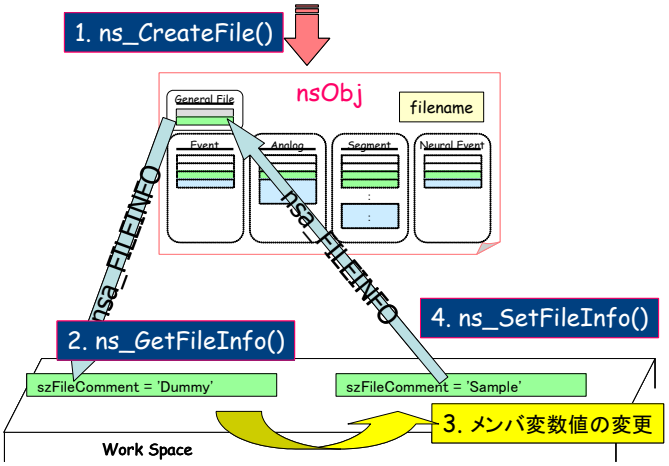


図 2 SampleData 作成フロー[1]

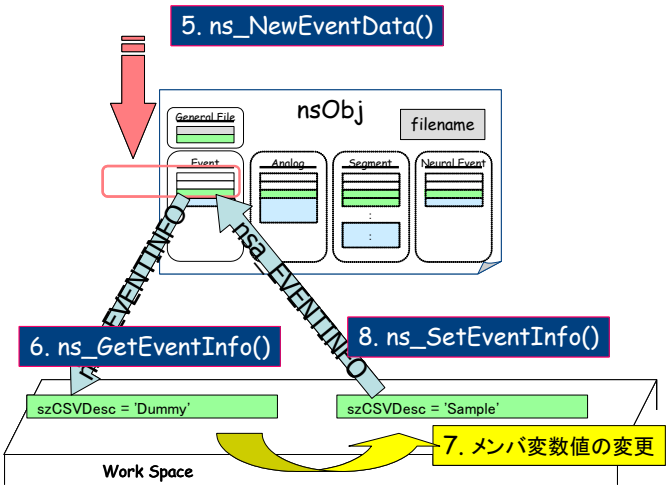


図 3 SampleData 作成フロー[2]

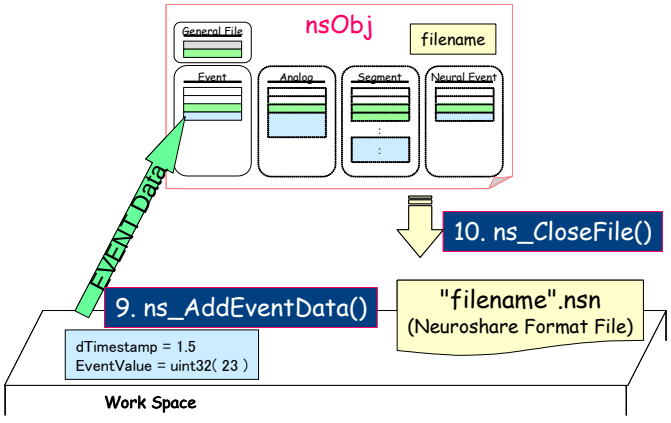


図 4 SampleData 作成フロー[3]

共通機能

ns_CreateFile クラスライブラリに共通する機能について記載しています。

オブジェクト操作禁止機能

この機能は、ns_Create クラスライブラリのオブジェクト nsObj の内容(つまり実験データの内容)を、
クラスライブラリの機能を介さずに、直接書き換えられないようにする機能です。

この機能により、Work Space から nsObj の内容を直接操作することを防ぎ、Neuroshare 形式から逸脱した値(以降誤値)が nsObj に格納されることを防ぎます。

なお、nsObj の内容を直接操作する要求が行われた場合、
その要求は禁じられている旨を、下記のようにメッセージとして出力します。

[オブジェクト操作禁止機能の動作例]

nsObj にある、ns_FILEINFO のメンバ変数を直接操作する要求を出した場合。

<< Work Space 上での入力 >> nsObj.ns_FILEINFO.szFileType = 'no meaning words';

【 動作結果 】

メッセージ ： ERROR : Can not edit a member of [nsObj] without using method

構造体の自動更新機能

この機能は、Entity の新規生成(ns_New***Data)(※1)や
Entity データの追加(ns_Add***Data)を行った際に、構造体と Entity データの間で整合性が維持されるよう、
生成/追加した内容に応じて構造体のメンバ変数の値を適切に更新するという機能です。

この機能により、構造体が Entity データと差異のある状況を防ぎ、矛盾する内容を含むファイルを作成されることを防ぎます。

なお、各メソッドが更新するメンバ変数とその値の一覧については、
<表 2 メソッドが更新するメンバ変数一覧>を参照してください。

(※1) Entity の新規生成(ns_New***Data)の場合、
構造体の初期化が実行された後、自動更新機能が実行されます。
初期値については<表 6 nsObj で管理するメンバ変数一覧>を参照してください。
初期化対象の構造体については各メソッドの解説を参照してください。

表 2 メソッドが更新するメンバ変数一覧

メソッド名	更新対象		更新値 ※但し(+*)は現在値に(*)追加するという意味	補足
	構造体	メンバ変数		
ns_NewEventData	ns_FILEINFO	dwEntityCount	+1	
	ns_TAGELEMENT	dwElemType	ns_ENTITY_EVENT	
		dwElemLength	180	構造体 ns_ENTITYINFO + ns_EVENTINFO の分
	ns_ENTITYINFO	szEntityLabel	[szEntityLabel の記載文字列]	New メソッド以降変わらない
		dwEntityType	ns_ENTITY_EVENT	
	ns_EVENTINFO	dwMinDataLength	$2^{32} - 1$	初期値を uint32 型の最大値とすることで、 正しい最小値を得る
		dwMaxDataLength	0	初期値を uint32 型の最小値とすることで、 正しい最大値を得る
ns_NewAnalogData	ns_FILEINFO	dwEntityCount	+1	
	ns_TAGELEMENT	dwElemType	ns_ENTITY_ANALOG	
		dwElemLength	304	構造体 ns_ENTITYINFO + ns_ANALOGINFO の分

	ns_ENTITYINFO	szEntityLabel	[szEntityLabel の記載文字列]	New メソッド以降変わらない
		dwEntityType	ns_ENTITY_ANALOG	
	ns_ANALOGINFO	dMinVal	$2^{63} - 1$	初期値を double 型の最大値とすることで、 正しい最小値を得る
		dMaxVal	-2^{63}	初期値を double 型の最小値とすることで、 正しい最大値を得る
ns_NewSegmentData	ns_FILEINFO	dwEntityCount	+1	
	ns_TAGELEMENT	dwElemType	ns_ENTITY_SEGMENT	
		dwElemLength	92	構造体 ns_ENTITYINFO + ns_SEGMENTINFO の分
	ns_ENTITYINFO	szEntityLabel	[szEntityLabel の記載文字列]	New メソッド以降変わらない
		dwEntityType	ns_ENTITY_SEGMENT	
	ns_SEGMENTINFO	dwMinSampleCount	$2^{32} - 1$	初期値を uint32 型の最大値とすることで、 正しい最小値を得る
		dwMaxSampleCount	0	初期値を uint32 型の最小値とすることで、 正しい最大値を得る
ns_NewNeuralEventData	ns_FILEINFO	dwEntityCount	+1	
	ns_TAGELEMENT	dwElemType	ns_ENTITY_NEURAL	
		dwElemLength	176	構造体 ns_ENTITYINFO + ns_NEURALINFO の分
	ns_ENTITYINFO	szEntityLabel	[szEntityLabel の記載文字列]	New メソッド以降変わらない
		dwEntityType	ns_ENTITY_NEURAL	
ns_AddEventData	ns_TAGELEMENT	dwElemLength	+12+[EventValue のバイト数]	
	ns_ENTITYINFO	dwItemCount	+1	
	ns_EVENTINFO	dwEventType	[ns_EVENT_***のいずれか]	データの内容に依存して更新される
		dwMinDataByteSize	[EventValue のバイト数最小値]	但し入力引数の値が、それまでの最小値を 下回る場合のみ更新される
		dwMaxDataByteSize	[EventValue のバイト数最大値]	但し入力引数の値が、それまでの最大値を 上回る場合のみ更新される
ns_AddAnalogData	ns_TAGELEMENT	dwElemLength	+12+8*[アナログデータの個数]	※[…]：入力引数 dData のベクトルの長さと一致
	ns_ENTITYINFO	dwItemCount	+[アナログデータの個数]	
	ns_ANALOGINFO	dMinVal	[アナログデータの最小値]	但し入力引数の値が、それまでの最小値を 下回る場合のみ更新される
		dMaxVal	[アナログデータの最大値]	但し入力引数の値が、それまでの最大値を 上回る場合のみ更新される
ns_AddSegmentData	ns_TAGELEMENT	dwElemLength	+264+8*[セグメントデータの個数]	追加する構造体 ns_SEGSOURCEINFO の分含む ※[…]：入力引数 dValue のベクトルの長さと一致

	ns_ENTITYINFO	dwItemCount	+1	
	ns_SEGMENTINFO	dwSourceCount	+1	
		dwMinSampleCount	[セグメントデータの個数最小値]	但し入力引数の値が、それまでの最小値を下回る場合のみ更新される
		dwMaxSampleCount	[セグメントデータの個数最大値]	但し入力引数の値が、それまでの最大値を上回る場合のみ更新される
	ns_SEGSOURCEINFO	dMinVal	[セグメントデータの最小値]	但し入力引数の値が、それまでの最小値を下回る場合のみ更新される
		dMaxVal	[セグメントデータの最大値]	但し入力引数の値が、それまでの最大値を上回る場合のみ更新される
ns_AddNeuralEventData	ns_TAGELEMENT	dwElemLength	+8	
	ns_ENTITYINFO	dwItemCount	+1	

入力値補正機能[double 自然数→uint32]

この機能は、ns_CreateFile クラスライブラリの各メソッドの、uint32 型の値の入力を求める箇所（※1）において、「double 型かつ、0 を含む自然数に変換可能な値(※2)」の入力を許すという機能です。

これにより、ユーザーは uint32 型の入力箇所において、0 を含む自然数と変換可能な値を入力すればよく、その値の型が uint32 か double かを意識する必要はありません。

MATLAB の仕様上、Work Space 上で任意に作成した数値型の変数は double 型として定義されてしまうため、ユーザーがわざわざ型変換(キャスト)をする必要がないよう、盛り込んでいます。

(※1)入力引数が uint32 型入力箇所である場合、
　　または入力引数の構造体に属するメンバ変数が uint32 型入力箇所である場合を指します。

(※2)「小数点以下が 0 であり、かつ 0 以上の数」と同意です。
　　[<double> 5.0]は[<uint32> 5]として扱われます。

入力値補正機能[char 文字数調整]

この機能は、ns_CreateFile クラスライブラリの各メソッドの、char[x]型の値の入力を求める箇所において、
char[x]に登録する文字列を、「ユーザーから入力された長さＡの文字列に(x-A)文字分の’ ’(ブランク)を付加した文字列」とする機能です。

ユーザーがわざわざ文字列に必要数’ ’(ブランク)を入力する必要がないよう、盛り込んでいます。

なお、A > x である(ユーザーが入力した文字列が入力制限文字数を超えた)場合、
char[x]に登録する文字列は、「ユーザーが入力した文字列の先頭から x 文字」となります。

ERROR 機能

この機能は、nsObj の内容に、各メソッドを介して誤値を登録できないようにし、
かつ Work Space 上にメソッド内の処理を行わなかった旨を表示する機能です。

この機能により、nsObj に誤値が登録されることを防ぎ、Neuroshare 形式に沿わないファイルが作成されてしまうことを防ぎます。

なお、この機能が搭載されている各メソッドは、どのような異常が発生したかという情報を元にして、
メソッドの呼び出し元には<表 3 ERROR 機能動作時エラーの内訳一覧>に記載されている戻り値[ns_Result]を返し、
かつ通知するメッセージとしては[ERROR メッセージ(抜粋)]に記載されている内容を出力します。

表 3 ERROR 機能動作時エラーの内訳一覧

ns_Result の値		ERROR メッセージ(抜粋)	エラーの内訳
ns_FILEERROR	-3	FILE MANIPULATION ERROR	ファイルの開閉または書き込みに失敗。
ns_WRONGLABEL	-101	WRONG DATA_TYPE :LABEL	出力ファイル名または Entity 名の型が不正。char 型ではない。
		WRONG NAME OF OUTPUT_FILE	出力ファイル名が不正。拡張子が'.nsn'以外で指定されている。
ns_WRONGID	-102	WRONG ID_TYPE	指定 ID の型が不正。uint32,または整数値に変換可能な double ではない。
		WRONG ID_VALUE	指定 ID の値が不正。ID で指す Entity 番号が存在しない。
ns_WRONGHEADER	-103	WRONG INFO :ns_****INFO :This is not correct structure	構造体の内容が不正。メンバ変数の項目数が正しい構造体の場合のそれよりも少ない。
		WRONG INFO :ns_****INFO.***(The Member doesn't exist)	構造体の内容が不正。未定義のメンバ変数や変更不可のメンバ変数が一つ以上含まれている。
ns_WRONGDATA	-104	WRONG DATA_TYPE :****Data :******	指定データの型が不正。正しい型で入力されていない。

[ERROR 機能の動作例]

メソッド ns_GetEventInfo の入力引数、EntityID 入力欄に小数値[3.2]が入っている場合。

<< Work Space 上での入力 >> [ns_Result , nsa_EVENTINFO] = ns_GetEventInfo(nsObj , 3.2);

[動作結果]

ns_Result の値 ： -102 (ns_WRONGID)

ERROR メッセージ ： WRONG ID_TYPE ***

nsObj の内容 ： 変更なし

WARNING 機能

この機能は、nsObj の内容に、各メソッドを介して誤値を登録できないようにし、
かつ Work Space 上にメソッドの処理を一部において行わなかった旨を表示する機能です。

ERROR 機能との違いは、「誤値の箇所は無視したが、正しい値が入力されている箇所は更新した」という部分です。
そのため、警告の意味としてメッセージを出力しますが、機能全体としては異常なく(戻り値が ns_OK として)完了します。

なお、この機能が搭載されている各メソッドは、どのような異常が発生したかという情報を元にして、
メソッドの呼び出し元には<表 4 WARNING 機能動作時警告の内訳一覧>に記載されている戻り値[ns_Result]を返し、かつ通知するメッセージとしては[WARNING メッセージ(抜粋)]に記載されている内容を出力します。

表 4 WARNING 機能動作時警告の内訳一覧

ns_Result の値		WARNING メッセージ(抜粋)	警告の内訳
ns_OK	0	WRONG INFO_TYPE : ns_****INFO.***	指定メンバ変数の型が不正。正しい型と一致しない。 ※例外：正しい型が uint32 であるメンバ変数に対しては、整数値に変換可能な double でも入力可能。
		WRONG INFO_VALUE : ns_****INFO.***	指定メンバ変数の値が不正。設定可能な範囲の値ではない。 ※設定可能範囲については<表 5 更新対象メンバ変数の設定可能範囲一覧>参照。

[WARNING 機能の動作例]

メソッド ns_SetFileInfo の使用により、ns_FILEINFO の構造体のメンバ変数 dwTime_Month と dwTime_Day に対してそれぞれ[12] [32]に設定する場合。([12]は設定可能な値だが、[32]は設定可能な値ではない)

<< Work Space 上での入力 >> [ns_Result , nsa_FILEINFO] = ns_GetFileInfo(nsObj);

<< Work Space 上での入力 >> nsa_FILEINFO.dwTime_Month = 12;

<< Work Space 上での入力 >> nsa_FILEINFO.dwTime_Day = 32;

<< Work Space 上での入力 >> [ns_Result , nsObj] = ns_SetFileInfo(nsObj , nsa_FILEINFO);

[動作結果]

ns_Result の値 ： 0 (ns_OK)

WARNING メッセージ ： WRONG INFO_VALUE ***

nsObj の内容 ： 変更あり [dwTime_Month は 12 に変化し、dwTime_Day は変化しない]

メソッド詳細

各メソッドが持つ機能を、処理の種別ごとに記載しています。

Neuroshare 形式ファイル作成処理

Neuroshare 形式ファイル作成処理に関連するメソッドとその処理概要、使用例等は以下の通りです。

[関連メソッド]

・ns_CreateFile　・・・　Neuroshare データ格納用オブジェクトの作成

- filename, sMagicCode, ns_FILEINFO の新規生成、初期化(※1)

【入力引数 filename の内容によっては ERROR 機能 – ns_WRONGLABEL が発生】

・ns_GetFileInfo　・・・　ファイル情報構造体の取得

- nsa_FILEINFO の取得(※2)

【入力引数 ID の内容によっては ERROR 機能 – ns_WRONGID が発生】

・ns_SetFileInfo　・・・　ファイル情報構造体の更新

- nsa_FILEINFO の更新(※2)

【入力引数 ID の内容によっては ERROR 機能 – ns_WRONGID が発生】

【入力引数 nsa_FILEINFO の内容によっては ERROR 機能 – ns_WRONGHEADER が発生】

【入力引数 nsa_FILEINFO の内容によっては WARNING 機能が発生】

・ns_CloseFile　・・・　全てのデータの統合

- 全ての中間ファイルを統合し、Neuroshare 形式のファイルを作成
- 全ての中間ファイルの削除

【中間ファイルの操作状態によっては ERROR 機能 – ns_FILEERROR が発生】

(※1)　初期値は<表 6 nsObj で管理するメンバ変数一覧>を参照してください。

(※2)　nsa_EVENTINFO 構造体のメンバ変数は
<表 6 nsObj で管理するメンバ変数一覧>の ns_FILEINFO の項目中、
[Get/Set***Info メソッドによる取得/更新の可/不可]が[可]であるもののみです。

[メソッド使用例] ※ >> 以下が Work Space における入力。

1. ns_CreateFile ・・・ Neuroshare データ格納用オブジェクトの作成

   >> [retA nsObj] = ns_CreateFile('dummy');

2. ns_GetFileInfo ・・・ ファイル情報構造体の取得

   >> [retB nsa_FILEINFO] = ns_GetFileInfo(nsObj);

3. [ユーザーによる構造体の内容の変更]

   >> nsa_FILEINFO.szFileComment = 'Words empty of meaning';

4. ns_SetFileInfo ・・・ ファイル情報構造体の更新

   >> [retC nsObj] = ns_SetFileInfo(nsObj, nsa_FILEINFO);

5. ＊＊＊＊＊ 各 Entity 登録処理 ＊＊＊＊＊

6. ns_CloseFile ・・・ 全てのデータの統合を行う

   >> [retD] = ns_CloseFile(nsObj);

[注意事項]

・ns_CreateFile メソッドの戻り値が ns_OK でない場合、以降他のメソッドを使用することができません。

[メソッド-Work Space 関連図]

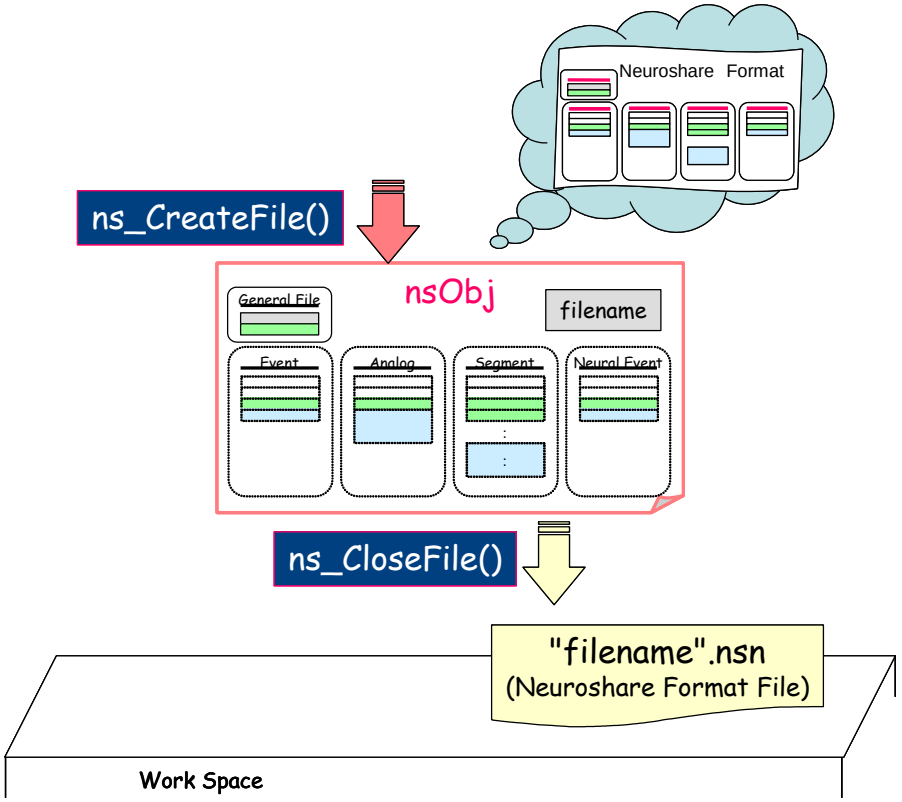


図 5 ns_CreateFile,ns_CloseFile

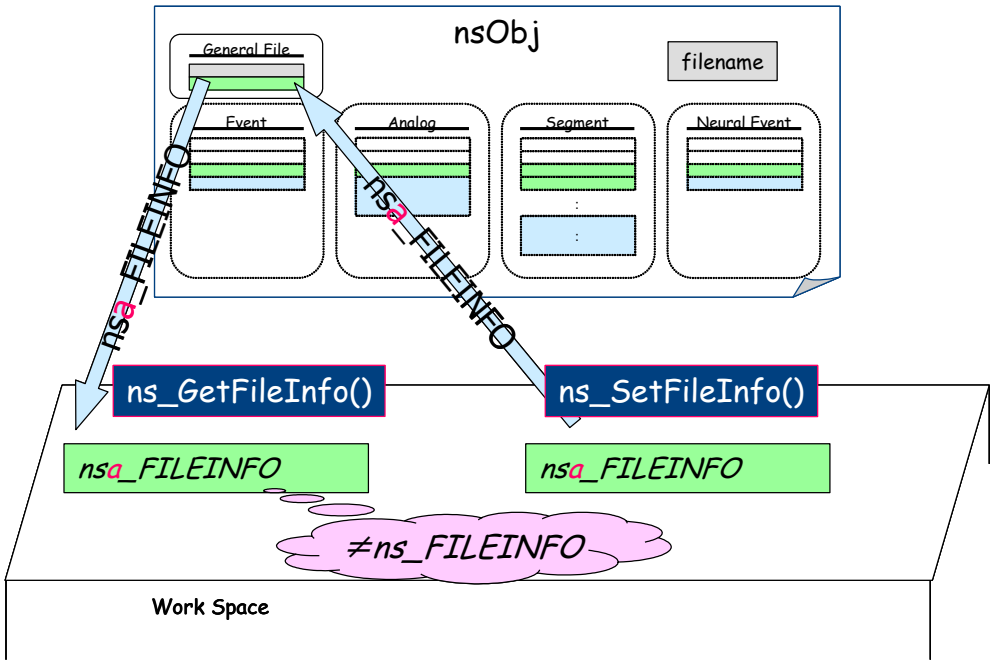


図 6 ns_GetFileInfo,ns_SetFileInfo

ns_CreateFile　Neuroshare データ格納用オブジェクトの作成を行う

[ns_Result nsObj] = ns_CreateFile(filename)

[動作概要]

ns_CreateFile()は、Neuroshare データ格納用オブジェクトを生成し、

生成結果(ns_Result) と Neuroshare データ格納用オブジェクト(nsObj)を返す

[入出力引数]

Input :

filename	– [char]	新規 Neuroshare 形式のファイルの名称(※1)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGLABEL
nsObj	– [struct]	Neuroshare データ格納用オブジェクト
		【ns_Result が ns_OK である場合】
		nsObj が格納される
		【ns_Result が ns_OK でない場合】
		''(ブランク)が格納される

[特記事項]

(※1)　filename には入力制限があります。

filename は'****.nsn'[拡張子が.nsn],または'****'[拡張子指定なし]の形でなくてはなりません。

filename が ns_CloseFile 使用後に完成する Neuroshare ファイルの名称となります。

尚、ファイルの出力先は、任意に選択可能です。

絶対パス、もしくはカレントディレクトリからみたパスで指定してください。

例 : file を C : ¥TEMP¥nsnConverter¥Data¥以下に作る場合

filename = 'C : ¥TEMP¥nsnConverter¥Data¥samplefilename.nsn' と設定してください。

ns_GetFileInfo　ファイル情報構造体の取得を行う

[ns_Result nsa_FILEINFO] = ns_GetFileInfo(nsObj)

[動作概要]

ns_GetFileInfo()は、nsObj で指定するファイル情報構造体(nsa_FILEINFO)を取得し、
取得結果(ns_Result) と ファイル情報構造体を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト

Output :

ns_Result	– [double]	ns_OK
nsa_FILEINFO	– [struct]	nsObj で特定する ns_FILEINFO

ns_SetFileInfo　ファイル情報構造体の更新を行う

[ns_Result nsObj] = ns_SetFileInfo(nsObj, nsa_FILEINFO)

[動作概要]

ns_SetFileInfo()は、nsObj で指定するファイル情報構造体を nsa_FILEINFO の内容に置き換え(=更新)、
更新結果(ns_Result) と 更新後の Neuroshare データ格納用オブジェクト(nsObj)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
nsa_FILEINFO	– [struct]	ファイル情報構造体(※1)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGHEADER
nsObj	– [struct]	Neuroshare データ格納用オブジェクト

【ns_Result が ns_OK である場合】
nsObj.ns_FILEINFO 構造体のメンバ変数の各値が
更新された nsObj が格納される

【ns_Result が ns_OK でない場合】
入力引数の nsObj がそのまま格納される

[特記事項]

(※1)　この構造体には、設定可能な範囲が定義されたメンバ変数が存在します。

設定可能な範囲の内訳については

<表 5 更新対象メンバ変数の設定可能範囲一覧>を参照してください。

表 5 更新対象メンバ変数の設定可能範囲一覧

更新対象		設定可能範囲	補足
構造体	メンバ変数	※[0…2]は"0 から 2 までの整数値"の意味	
ns_FILEINFO	dwTime_Month	[1…12]	1-12 月
	dwTime_DayOfWeek	[0…6]	曜日 0:日曜 6:土曜
	dwTime_Day	[1…31]	1-31 日
	dwTime_Hour	[0…23]	0-23 時
	dwTime_Min	[0…59]	0-59 分
	dwTime_Sec	[0…59]	0-59 秒
	dwTime_MilliSec	[0…1000]	0-1000msec

ns_CloseFile 全てのデータの統合を行う

ns_Result = ns_CloseFile(nsObj)

[動作概要]

ns_CloseFile()は、Neuroshare データ格納用オブジェクトと、各 Entity で作成した Entity データの情報を統合し、
Neuroshare ファイルを作成(※1)する。中間ファイルを全て削除(※2)した後、
ファイル統合結果(ns_Result)を返す。

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト

Output :

ns_Result	– [double]	ns_OK または ns_FILEERROR

[特記事項]

(※1) Neuroshare 形式ファイルには、
Event→Analog→Segment→NeuralEvent の Entity 種別順で書き込まれます。
例えば Work Space 上において Entity が Event,NeuralEvent,Analog(1),Analog(2)の順序で
作成された場合でも、Event,Analog(1),Analog(2),NeuralEvent の順で Entity を作成した
Neuroshare 形式のファイルが作成されます。
<注意事項 : Ver. 1.0>を参照してください。

(※2) Entity のデータが格納されている中間ファイルは不要のため削除します。

Event Entity 登録処理

Event Entity 登録処理に関連するメソッドとその処理概要、使用例等は以下の通りです。

[関連メソッド]

・ns_NewEventData　・・・　新規イベントデータの生成

- ns_TAGELEMENT, ns_ENTITYINFO, ns_EVENTINFO の新規生成、初期化(※1)
- イベントデータを識別する ID の新規生成(※2)
- イベントデータを格納する中間ファイルの作成
- 構造体の自動更新(※3)

【入力引数 szEntityLabel の内容によっては ERROR 機能 － ns_WRONGLABEL が発生】

・ns_GetEventInfo　・・・　イベント情報構造体の取得

- nsa_EVENTINFO の取得(※4)

【入力引数 ID の内容によっては ERROR 機能 － ns_WRONGID が発生】

・ns_SetEventInfo　・・・　イベント情報構造体の更新

- nsa_EVENTINFO の更新(※4)

【入力引数 ID の内容によっては ERROR 機能 － ns_WRONGID が発生】

【入力引数 nsa_EVENTINFO の内容によっては ERROR 機能 － ns_WRONGHEADER が発生】

【入力引数 nsa_EVENTINFO の内容によっては WARNING 機能が発生】

・ns_AddEventData　・・・　イベントデータの追加

- 中間ファイルへの dTimestamp, dwDataByteSize, EventValue の書き込み
- 構造体の自動更新(※3)

【入力引数 ID の内容によっては ERROR 機能 － ns_WRONGID が発生】

【入力引数 dTimestamp,EventValue の内容によっては ERROR 機能 － ns_WRONGDATA が発生】

(※1)　初期値は<表 6 nsObj で管理するメンバ変数一覧>を参照してください。

(※2)　この ID 値を元に Get,Set,Add メソッドを使用し、意図した Event Entity に対する操作を行います。

(※3)　更新対象と更新値は<表 2 メソッドが更新するメンバ変数一覧>を参照してください。

(※4)　nsa_EVENTINFO 構造体のメンバ変数は
<表 6 nsObj で管理するメンバ変数一覧>の ns_EVENTINFO の項目中、
[Get/Set***Info メソッドによる取得/更新の可/不可]が[可]であるもののみです。

[メソッド使用例] ※ >> 以下が Work Space における入力。nsObj 作成後に使用可能。

1. ns_NewEventData ・・・ **新規イベントデータの生成**

   >> [retD nsObj EventID] = ns_NewEventData(nsObj, 'dummy');

2. ns_GetEventInfo ・・・ **イベント情報構造体の取得**

   >> [retE nsa_EVENTINFO] = ns_GetEventInfo(nsObj, EventID);

3. **[ユーザーによる構造体の内容の変更]**

   >> nsa_EVENTINFO.szCSVDesc = 'Words empty of meaning';

4. ns_SetEventInfo ・・・ **イベント情報構造体の更新**

   >> [retF nsObj] = ns_SetEventInfo(nsObj, EventID, nsa_EVENTINFO);

5. ns_AddEventData ・・・ **イベントデータの追加**

   >> [retG nsObj] = ns_AddEventData(nsObj, EventID, 1.5, uint32(23));

[注意事項]

・CSV 形式の EventValue には対応していません。詳しくは<Ver. 1.0>を参照してください。

[メソッド−Work Space 関連図]

※自動更新対象は記載省略

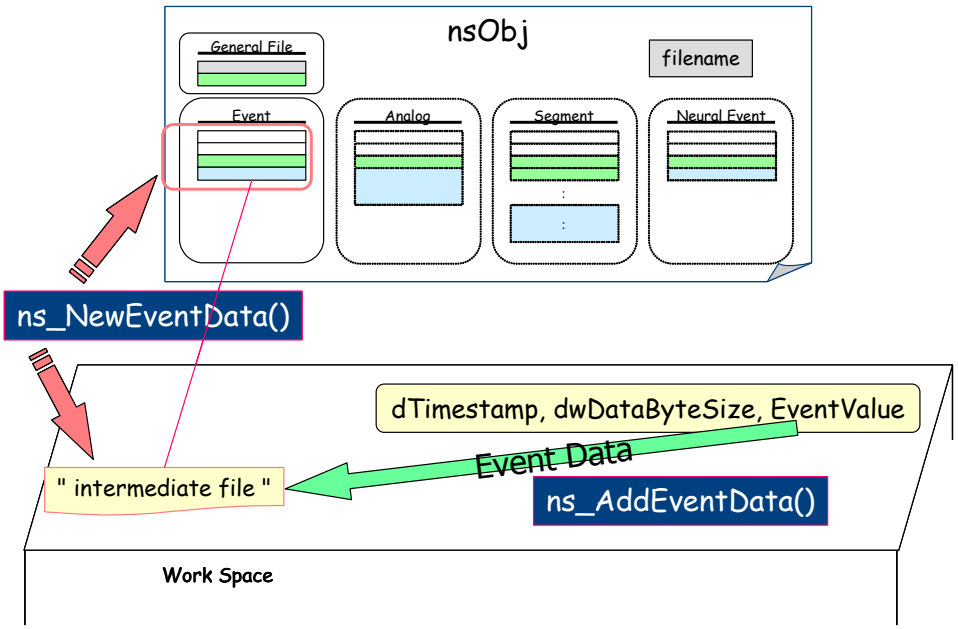


図 7 ns_NewEventData,ns_AddEventData

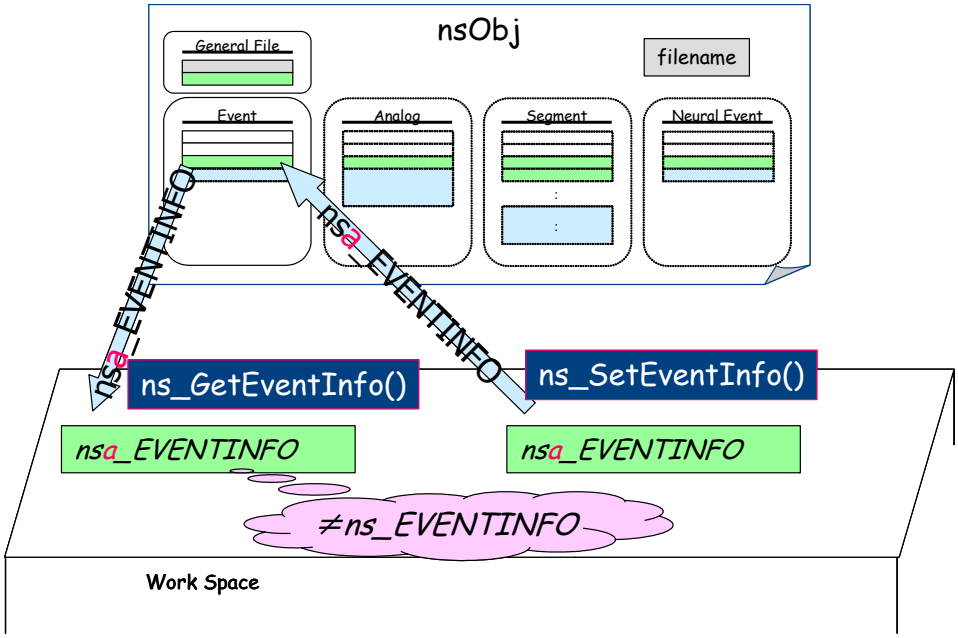


図 8 ns_GetEventInfo,ns_SetEventInfo

ns_NewEventData　新規イベントデータの生成処理を行う

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

[動作概要]

ns_NewEventData()は、新規イベントデータの格納先（中間ファイル）を生成し、
生成結果(ns_Result)，生成処理後の Neuroshare データ格納用オブジェクト(nsObj)，
生成したイベントデータを識別するための ID を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
szEntityLabel	– [char]	<任意入力項目>新規イベントデータの Entity 名(※1)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGLABEL
nsObj	– [struct]	Neuroshare データ格納用オブジェクト 【ns_Result が ns_OK である場合】 nsObj の各値が変更された nsObj(※2)が格納される 【ns_Result が ns_OK でない場合】 入力引数の nsObj がそのまま格納される
ID	– [uint32]	生成したイベントデータを識別するための ID 【ns_Result が ns_OK である場合】 新規 ID 値が格納される(※3) 【ns_Result が ns_OK でない場合】 ''(ブランク)が格納される

[特記事項]

(※1)　Entity 名の入力は任意です。

Entity 名を省略した(第 2 入力引数が存在しない)場合、これは自動的に''(ブランク)となります。

(※2)　nsObj.Event[ID].FID に「作成した中間ファイルへのポインタ」が格納され、

自動更新も実施されています。

尚、ここで作成された中間ファイルは、ファイル統合(ns_CloseFile)後、自動的に削除されます。

(※3)　Event Entity の登録順にみて何番目か、という情報が入ります。

他の Entity 種別の登録数とは無関係です。

ns_GetEventInfo　イベント情報構造体の取得を行う

[ns_Result nsa_EVENTINFO] = ns_GetEventInfo(nsObj, ID)

[動作概要]

ns_GetEventInfo()は、nsObj と ID で指定するイベント情報構造体(nsa_EVENTINFO)を取得し、
取得結果(ns_Result) と イベント情報構造体を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
ID	– [uint32]	イベントデータを識別するための ID

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID
nsa_EVENTINFO	– [struct]	イベント情報構造体

【ns_Result が ns_OK である場合】
nsObj, ID で特定される、
nsa_EVENTINFO 構造体が格納される

【ns_Result が ns_OK でない場合】
''(ブランク)が格納される

ns_SetEventInfo　イベント情報構造体の更新を行う

[ns_Result nsObj] = ns_SetEventInfo(nsObj, ID, nsa_EVENTINFO)

[動作概要]

ns_SetEventInfo()は、nsObj と ID で指定するイベント情報構造体を nsa_EVENTINFO の内容に置き換え(=更新)、更新結果(ns_Result) と 更新後の Neuroshare データ格納用オブジェクト(nsObj)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
ID	– [uint32]	イベントデータを識別するための ID
nsa_EVENTINFO	– [struct]	イベント情報構造体

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID または ns_WRONGHEADER
nsObj	– [struct]	Neuroshare データ格納用オブジェクト 【ns_Result が ns_OK である場合】 nsObj.Event{ID}.ns_EVENTINFO 構造体のメンバ変数の各値が更新された nsObj が格納される 【ns_Result が ns_OK でない場合】 入力引数の nsObj がそのまま格納される

ns_AddEventData　イベントデータの追加を行う

[ns_Result nsObj] = ns_AddEventData(nsObj, ID, dTimestamp, EventValue)

[動作概要]

ns_AddEventData()は、nsObj と ID で指定するイベントデータに、

dTimestamp,EventValue で表現するデータを追加し(※1)、

追加結果(ns_Result) と 追加後の Neuroshare データ格納用オブジェクト(nsObj)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
ID	– [uint32]	イベントデータを識別するための ID
dTimestamp	– [double]	タイムスタンプ値(scalar : 1×1)
EventValue	– [(u)int8, (u)int16, (u)int32 or char]	イベント値(※2)(※3)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID または ns_WRONGDATA
nsObj	– [struct]	Neuroshare データ格納用オブジェクト

【ns_Result が ns_OK である場合】

nsObj の各値が更新された nsObj が格納される

【ns_Result が ns_OK でない場合】

入力引数の nsObj がそのまま格納される

[特記事項]

(※1)　Event Data として、ns_NewEventData で作成した nsObj.Event{ID}.FID で指す中間ファイルに、
下記順序で Entity データを書き込みます。

1. dTimestamp
2. dwDataByteSize[EventValue のバイト数, (EventValue の内容を元に作成) scalar : 1×1]
3. EventValue

(※2)　EventValue は、同一 EventID 中において同じ型をとらなければなりません。
[ただし、uint**型と int**型(**は同じ値)の共存は認めます。]
それに反した場合はエラーとなります。
例. 初回の ns_AddEventData 使用時に uint32 の値を入力した場合、
n 回目の ns_AddEventData 使用時において uint32 or int32 以外の型をもつ値の入力はエラーとなる。

(※3)　イベント値として、char(文字列)の入力を許可しています。
ns_AddEventData(nsObj, ID, 'No_Meaning') という使い方が可能です。
一方で CSV 形式の入力をイベント値として許可しません。

Analog Entity 登録処理

Analog Entity 登録処理に関連するメソッドとその処理概要、使用例等は以下の通りです。

[関連メソッド]

・ns_NewAnalogData　・・・　新規アナログデータの生成

- ns_TAGELEMENT, ns_ENTITYINFO, ns_ANALOGINFO の新規生成、初期化(※1)
- アナログデータを識別する ID の新規生成(※2)
- アナログデータを格納する中間ファイルの作成
- 構造体の自動更新(※3)

【入力引数 szEntityLabel の内容によっては ERROR 機能 – ns_WRONGLABEL が発生】

・ns_GetAnalogInfo　・・・　アナログ情報構造体の取得

- nsa_ANALOGINFO の取得(※4)

【入力引数 ID の内容によっては ERROR 機能 – ns_WRONGID が発生】

・ns_SetAnalogInfo　・・・　アナログ情報構造体の更新

- nsa_ANALOGINFO の更新(※4)

【入力引数 ID の内容によっては ERROR 機能 – ns_WRONGID が発生】

【入力引数 nsa_ANALOGINFO の内容によっては ERROR 機能 – ns_WRONGHEADER が発生】

【入力引数 nsa_ANALOGINFO の内容によっては WARNING 機能が発生】

・ns_AddAnalogData　・・・　アナログデータの追加

- 中間ファイルへの dTimestamp, dwDataCount, dData の書き込み
- 構造体の自動更新(※3)

【入力引数 ID の内容によっては ERROR 機能 – ns_WRONGID が発生】

【入力引数 dTimestamp,dData の内容によっては ERROR 機能 – ns_WRONGDATA が発生】

(※1)　初期値は<表 6 nsObj で管理するメンバ変数一覧>を参照してください。

(※2)　この ID 値を元に Get,Set,Add メソッドを使用し、意図した Analog Entity に対する操作を行います。

(※3)　更新対象と更新値は<表 2 メソッドが更新するメンバ変数一覧>を参照してください。

(※4)　nsa_ANALOGINFO 構造体のメンバ変数は、ns_ANALOGINFO のそれと同一です。

特に、ns_ANALOGINFO 構造体のメンバ変数 dMin/MaxVal を、任意の値に設定することが可能です。

詳しくは<nsa_***INFO 構造体とは>を参照してください。

[メソッド使用例] ※ >> 以下が Work Space における入力。nsObj 作成後に使用可能。

1. ns_NewAnalogData ・・・ **新規アナログデータの生成**

   >> [retI nsObj AnalogID] = ns_NewAnalogData(nsObj, 'dummy');

2. ns_GetAnalogInfo ・・・ **アナログ情報構造体の取得**

   >> [retJ nsa_ANALOGINFO] = ns_GetAnalogInfo(nsObj, AnalogID);

3. **[ユーザーによる構造体の内容の変更]**

   >> nsa_ANALOGINFO.szProbeInfo = 'Words empty of meaning';

4. ns_SetAnalogInfo ・・・ **アナログ情報構造体の更新**

   >> [retK nsObj] = ns_SetAnalogInfo(nsObj, AnalogID, nsa_ANALOGINFO);

5. ns_AddAnalogData ・・・ **アナログデータの追加**

   >> [retL nsObj] = ns_AddAnalogData(nsObj, AnalogID, 1.5, [5.6 4.5 3.4]);

[注意事項]

・特にありません。

[メソッド-Work Space 関連図]

※自動更新対象は記載省略

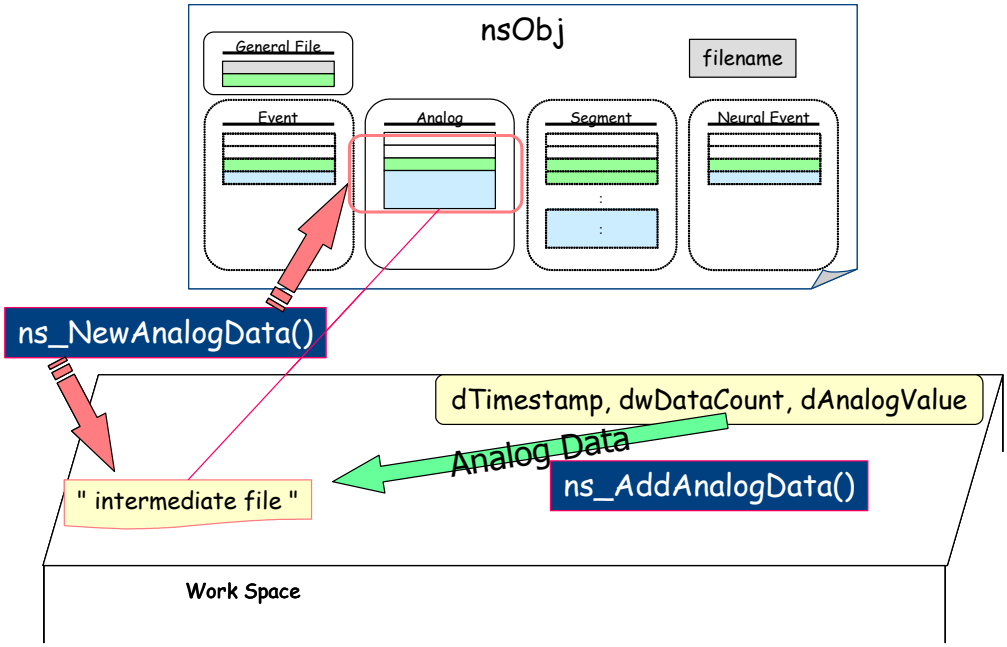


図 9 ns_NewAnalogData,ns_AddAnalogData

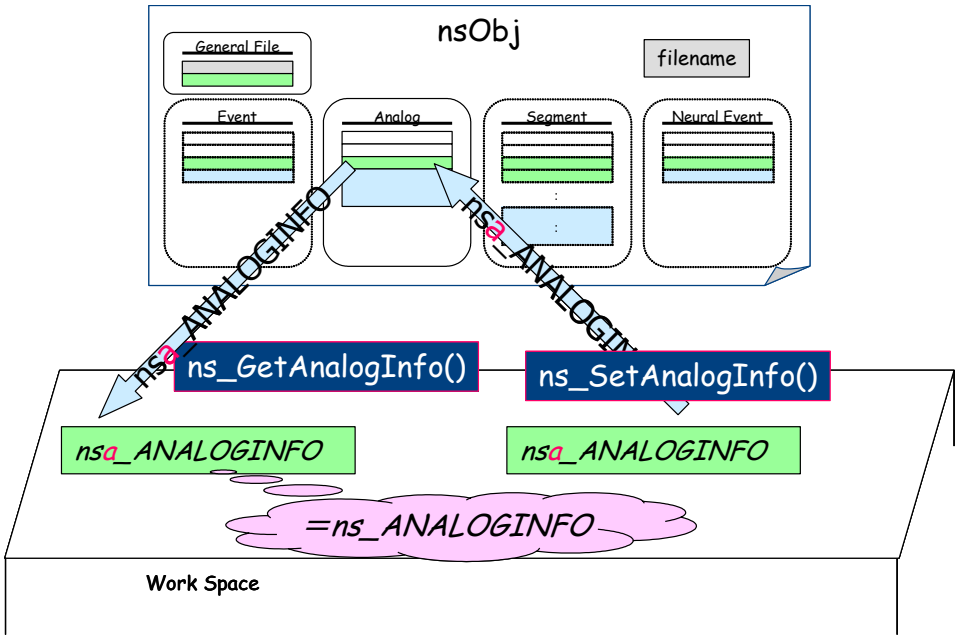


図 10 ns_GetAnalogInfo,ns_SetAnalogInfo

ns_NewAnalogData　新規アナログデータの生成処理を行う

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

[動作概要]

ns_NewAnalogData()は、新規アナログデータの格納先(中間ファイル)を生成し、
生成結果(ns_Result)，生成処理後の Neuroshare データ格納用オブジェクト(nsObj)，
生成したアナログデータを識別するための ID を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
szEntityLabel	– [char]	<任意入力項目>新規アナログデータの Entity 名(※1)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGLABEL
nsObj	– [struct]	Neuroshare データ格納用オブジェクト 【ns_Result が ns_OK である場合】 nsObj の各値が更新された nsObj(※2)が格納される 【ns_Result が ns_OK でない場合】 入力引数の nsObj がそのまま格納される
ID	– [uint32]	生成したアナログデータを識別するための ID 【ns_Result が ns_OK である場合】 新規 ID 値が格納される(※3) 【ns_Result が ns_OK でない場合】 ''(ブランク)が格納される

[特記事項]

(※1)　Entity 名の入力は任意です。

Entity 名を省略した(第 2 入力引数が存在しない)場合、これは自動的に''(ブランク)となります。

(※2)　nsObj.Analog[ID].FID に「作成した中間ファイルへのポインタ」が格納され、

自動更新も実施されています。

尚、ここで作成された中間ファイルは、ファイル統合(ns_CloseFile)後、自動的に削除されます。

(※4)　Analog Entity の登録順にみて何番目か、という値が入ります。

他の Entity 種別の登録数とは無関係です。

ns_GetAnalogInfo　アナログ情報構造体の取得を行う

[ns_Result nsa_ANALOGINFO] = ns_GetAnalogInfo(nsObj, ID)

[動作概要]

ns_GetAnalogInfo()は、nsObj と ID で指定するアナログ情報構造体(nsa_ANALOGINFO)を取得し、
取得結果(ns_Result) と アナログ情報構造体を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
ID	– [uint32]	アナログデータを識別するための ID

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID
nsa_ANALOGINFO	– [struct]	アナログ情報構造体

【ns_Result が ns_OK である場合】
nsObj,ID で特定される、
nsa_ANALOGINFO 構造体が格納される

【ns_Result が ns_OK でない場合】
''(ブランク)が格納される

ns_SetAnalogInfo　アナログ情報構造体の更新を行う

[ns_Result nsObj] = ns_SetAnalogInfo(nsObj, ID, nsa_ANALOGINFO)

[動作概要]

ns_SetAnalogInfo()は、nsObj と ID で指定するアナログ情報構造体を
nsa_ANALOGINFO の内容に置き換え(=更新)、更新結果(ns_Result) と
更新後の Neuroshare データ格納用オブジェクト(nsObj)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
ID	– [uint32]	アナログデータを識別するための ID
nsa_ANALOGINFO	– [struct]	アナログ情報構造体

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID または ns_WRONGHEADER
nsObj	– [struct]	Neuroshare データ格納用オブジェクト

【ns_Result が ns_OK である場合】
nsObj.Analog{ID}.ns_ANALOGINFO 構造体のメンバ変数の
各値が更新された nsObj が格納される

【ns_Result が ns_OK でない場合】
入力引数の nsObj がそのまま格納される

ns_AddAnalogData　アナログデータの追加を行う

[ns_Result nsObj] = ns_AddAnalogData(nsObj, ID, dTime, dData)

[動作概要]

ns_AddAnalogData()は、nsObj と ID で指定するアナログデータに、

dTime,dData で表現するデータを追加し(※1)、

追加結果(ns_Result) と 追加後の Neuroshare データ格納用オブジェクト(nsObj)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
ID	– [uint32]	アナログデータを識別するための ID
dTime	– [double]	タイムスタンプ値(scalar : 1×1)
dData	– [double]	アナログ値(vector : 1×n)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID または ns_WRONGDATA
nsObj	– [struct]	Neuroshare データ格納用オブジェクト 【ns_Result が ns_OK である場合】 nsObj の各値が更新された nsObj が格納される 【ns_Result が ns_OK でない場合】 入力引数の nsObj がそのまま格納される

[特記事項]

(※1)　Analog Data として、ns_NewAnalogData で作成した nsObj.Analog{ID}.FID で指す中間ファイルに、
下記順序で Entity データを書き込みます。

1. dTime (dTimestamp として)
2. dwDataCount[dData(double : 1×n)のデータ数n, (dData の内容を元に作成) scalar : 1×1]
3. dData (dAnalogValue[0]・・・dAnalogValue[dwDataCount −1] として)

Segment Entity 登録処理

Segment Entity 登録処理に関連するメソッドとその処理概要、使用例等は以下の通りです。

[関連メソッド]

・ns_NewSegmentData ・・・ 新規セグメントデータの生成

- ns_TAGELEMENT, ns_ENTITYINFO, ns_SEGMENTINFO の新規生成、初期化(※1)
- セグメントデータを識別する ID の新規生成(※2)
- セグメントデータを格納する中間ファイルの作成
- セグメントソース情報構造体を識別する ID の新規作成(※7)
- セグメントソース情報構造体の追加、初期化(※7)
- 構造体の自動更新(※3)

【入力引数 szEntityLabel の内容によっては ERROR 機能 － ns_WRONGLABEL が発生】

・ns_GetSegmentInfo ・・・ セグメント情報構造体の取得

- nsa_SEGMENTINFO の取得(※4)

【入力引数 SEGID の内容によっては ERROR 機能 － ns_WRONGID が発生】

・ns_GetSegmentSourceInfo ・・・ セグメントソース情報構造体の取得

- nsa_SEGSOURCEINFO の取得(※5)

【入力引数 SEGID の内容によっては ERROR 機能 － ns_WRONGID が発生】

【入力引数 SEGSOURCEID の内容によっては ERROR 機能 － ns_WRONGID が発生】

・ns_SetSegmentInfo ・・・ セグメント情報構造体の更新

- nsa_SEGMENTINFO の更新(※4)

【入力引数 SEGID の内容によっては ERROR 機能 － ns_WRONGID が発生】

【入力引数 nsa_SEGMENTINFO の内容によっては ERROR 機能 － ns_WRONGHEADER が発生】

【入力引数 nsa_SEGMENTINFO の内容によっては WARNING 機能が発生】

・ns_SetSegmentSourceInfo ・・・ セグメントソース情報構造体の更新

- nsa_SEGSOURCEINFO の更新(※5)

【入力引数 SEGID の内容によっては ERROR 機能 － ns_WRONGID が発生】

【入力引数 SEGSOURCEID の内容によっては ERROR 機能 － ns_WRONGID が発生】

【入力引数 nsa_NEURALINFO の内容によっては ERROR 機能 － ns_WRONGHEADER が発生】

【入力引数 nsa_NEURALINFO の内容によっては WARNING 機能が発生】

・ns_AddSegmentData　　・・・　　~~セグメントソース情報構造体の追加と、~~(※7)セグメントデータの追加

~~・　セグメントソース情報構造体を識別する ID の新規作成(※6)~~

~~・　セグメントソース情報構造体の追加、初期化(※1)~~

・　中間ファイルへの dwSampleCount, dTimestamp, dwUnitID, dValue の書き込み

・　構造体の自動更新(※3)

【入力引数 SEGID の内容によっては ERROR 機能 － ns_WRONGID が発生】

【入力引数 dTimestamp, dwUnitID, dValue の内容によっては ERROR 機能 － ns_WRONGDATA が発生】

(※1)　初期値は<表 6 nsObj で管理するメンバ変数一覧>を参照してください。

(※2)　この ID 値を元に GetSegmentInfo,SetSegmentInfo,AddSegmentData メソッドを使用し、意図した Segment Entity に対する操作を行います。

(※3)　更新対象と更新値は<表 2 メソッドが更新するメンバ変数一覧>を参照してください。

(※4)　nsa_SEGMENTINFO 構造体のメンバ変数は
<表 6 nsObj で管理するメンバ変数一覧>の ns_SEGMENTINFO の項目中、
[Get/Set***Info メソッドによる取得/更新の可/不可]が[可]であるもののみです。

(※5)　nsa_SEGSOURCEINFO 構造体のメンバ変数は、ns_SEGSOURCEINFO のそれと同一です。

(※6)　この ID 値を元に GetSegmentSourceInfo,SetSegmentSourceInfo メソッドを使用し、意図した Segment Entity の ns_SEGSOURCEINFO に対する操作を行います。

(※7)　Ver. 1.3 － SegmentEntity の仕様変更(1Segment Entity あたりの SegSourceInfo ヘッダーの登録数制限化)により、ns_SEGSOURCEINFO 作成のタイミングは ns_AddSegmentData()から ns_NewSegmentData に変更されました。

[メソッド使用例] ※ >> 以下が Work Space における入力。nsObj 作成後に使用可能。

1. ns_NewSegmentData ・・・ **新規セグメントデータの生成**

   >> [retM nsObj SEGID] = ns_NewSegmentData(nsObj, 'dummy');

2. ns_GetSegmentInfo ・・・ **セグメント情報構造体の取得**

   >> [retN nsa_SEGMENTINFO] = ns_GetSegmentInfo(nsObj, SEGID);

3. **[ユーザーによる構造体の内容の変更]**

   >> nsa_SEGMENTINFO.szUnits = 'Words empty of meaning';

4. ns_SetSegmentInfo ・・・ **セグメント情報構造体の更新**

   >> [retO nsObj] = ns_SetSegmentInfo(nsObj, SEGID, nsa_SEGMENTINFO);

5. ns_GetSegmentSourceInfo ・・・ **セグメントソース情報構造体の取得**

   >> [retQ nsa_SEGSOURCEINFO] = ns_GetSegmentSourceInfo(nsObj, SEGID, SEGSOURCEID);

6. **[ユーザーによる構造体の内容の変更]**

   >> nsa_SEGSOURCEINFO.szProbeInfo = 'Words empty of meaning';

7. ns_SetSegmentSourceInfo ・・・ **セグメントソース情報構造体の更新**

   >> [retR nsObj] = ns_SetSegmentSourceInfo(nsObj, SEGID, SEGSOURCEID, nsa_SEGSOURCEINFO);

8. ns_AddSegmentData ・・・ **セグメントデータの追加**

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

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

[注意事項]

~~・SEGSOURCEID は、ns_AddSegmentData メソッドで生成されるため、~~
~~同一の SEGSOURCEID において Add メソッドの使用以前に~~
~~ns_Get/SetSegmentSourceInfo メソッドを使用することはできません。~~

・SEGSOURCEID は、ns_NewSegmentData メソッドで 1 つのみ生成され、かつ個数は維持されます。[Ver. 1.3 仕様] そのため、SEGSOURCEID は実質[1]しか取りえず、それ以外の値を入力すると、SEGSOURCEINFO が存在しない旨が表示されることになります。

[メソッド-Work Space 関連図]

※自動更新対象は記載省略

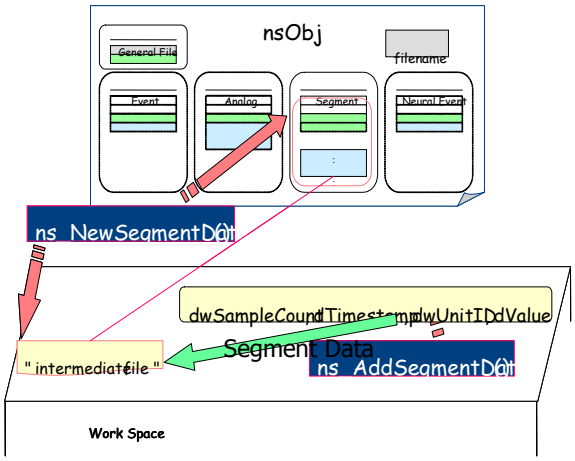


図 11 ns_NewSegmentData,ns_AddSegmentData

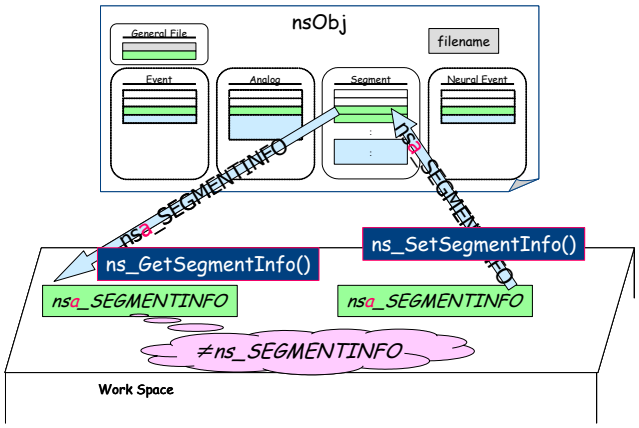


図 12 ns_GetSegmentInfo,ns_SetSegmentInfo

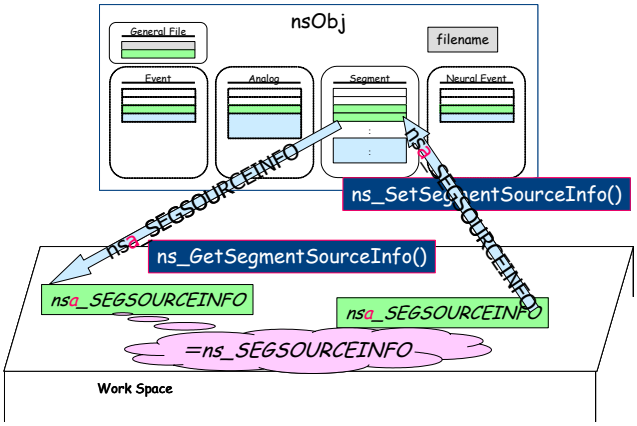


図 13 ns_GetSegmentSourceInfo,ns_SetSegmentSourceInfo

ns_NewSegmentData　新規セグメントデータの生成処理を行う

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

[動作概要]

ns_NewSegmentData()は、nsObj に新規セグメントデータの格納先(中間ファイル)を生成し、
生成結果(ns_Result)，生成処理後の Neuroshare データ格納用オブジェクト(nsObj(※4))，
生成したセグメントデータを識別するための ID(SEGID)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
szEntityLabel	– [char]	<任意入力項目>新規セグメントデータの Entity 名(※1)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGLABEL
nsObj	– [struct]	Neuroshare データ格納用オブジェクト 【ns_Result が ns_OK である場合】 nsObj の各値が更新された nsObj(※2)が格納される 【ns_Result が ns_OK でない場合】 入力引数の nsObj がそのまま格納される
SEGID	– [uint32]	生成したセグメントデータを識別するための ID 【ns_Result が ns_OK である場合】 新規 ID 値が格納される(※3) 【ns_Result が ns_OK でない場合】 ''(ブランク)が格納される

[特記事項]

（※1）　Entity 名の入力は任意です。

Entity 名を省略した(第 2 入力引数が存在しない)場合、これは自動的に''(ブランク)となります。

（※2）　nsObj.Segment[SEGID].FID に「作成した中間ファイルへのポインタ」が格納され、

自動更新も実施されています。

尚、ここで作成された中間ファイルは、ファイル統合(ns_CloseFile)後、自動的に削除されます。

（※3）　SegmentEntity の登録順にみて何番目か、という情報が入ります。

他の Entity 種別の登録数とは無関係です。

（※4）　Ver1.3 － SegmentEntity の仕様変更(1Segment Entity あたりの SegSourceInfo ヘッダーの登録数制限化）

により、本メソッドの機能が以下のように変更されました。

旧

1.　ns_SEGSOURCEINFO を作成しない。

新

1.　ns_SEGSOURCEINFO を作成して追加する。

ns_GetSegmentInfo　セグメント情報構造体の取得を行う

[ns_Result nsa_SEGMENTINFO] = ns_GetSegmentInfo(nsObj, SEGID)

[動作概要]

ns_GetSegmentInfo()は、nsObj と ID で指定するセグメント情報構造体(nsa_SEGMENTINFO)を取得し、
取得結果(ns_Result) と セグメント情報構造体を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
SEGID	– [uint32]	セグメントデータを識別するための ID

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID
nsa_SEGMENTINFO	– [struct]	セグメント情報構造体

【ns_Result が ns_OK である場合】
nsObj,SEGID で特定される、
ns_SEGMENTINFO 構造体が格納される

【ns_Result が ns_OK でない場合】
''(ブランク)が格納される

ns_GetSegmentSourceInfo　セグメントソース情報構造体の取得を行う

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

[動作概要]

ns_GetSegmentSourceInfo()は、nsObj, SEGID, SEGSOURCEID で指定する
セグメントソース情報構造体(nsa_SEGSOURCEINFO)を取得し、
取得結果(ns_Result) と セグメントソース情報構造体を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
SEGID	– [uint32]	セグメントデータを識別するための ID
SEGSOURCEID	– [uint32]	セグメントソース情報構造体を識別するための ID

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID
nsa_SEGSOURCEINFO	– [struct]	セグメントソース情報構造体

【ns_Result が ns_OK である場合】
nsObj, SEGID, SEGSOURCEID で特定される、
nsa_SEGSOURCEINFO 構造体が格納される

【ns_Result が ns_OK でない場合】
''(ブランク)が格納される

ns_SetSegmentInfo　セグメント情報構造体の更新を行う

[ns_Result nsObj] = ns_SetSegmentInfo(nsObj, SEGID, nsa_SEGMENTINFO)

[動作概要]

ns_SetSegmentInfo()は、nsObj と SEGID で指定するセグメント情報構造体を
nsa_SEGMENTINFO の内容に置き換え(=更新)、更新結果(ns_Result) と
更新後の Neuroshare データ格納用オブジェクト(nsObj)を返す
<図 12 ns_GetSegmentInfo,ns_SetSegmentInfo>を参照

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
SEGID	– [uint32]	セグメントデータを識別するための ID
nsa_SEGMENTINFO	– [struct]	セグメント情報構造体(※1)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID または ns_WRONGHEADER
nsObj	– [struct]	Neuroshare データ格納用オブジェクト

【ns_Result が ns_OK である場合】
nsObj.Segment{SEGID}.ns_SEGMENTINFO 構造体の
メンバ変数の各値が更新された nsObj が格納される

【ns_Result が ns_OK でない場合】
入力引数の nsObj がそのまま格納される

ns_SetSegmentSourceInfo　セグメントソース情報構造体の更新を行う

[ns_Result nsObj] = ns_SetSegmentSourceInfo(nsObj, SEGID, SEGSOURCEID, nsa_SEGSOURCEINFO)

[動作概要]

ns_SetSegmentSourceInfo()は、nsObj と SEGID,SEGSOURCEID で指定するセグメントソース情報構造体を nsa_SEGSOURCEINFO の内容に置き換え(=更新)、
更新結果(ns_Result) と 更新後の Neuroshare データ格納用オブジェクト(nsObj)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
SEGID	– [uint32]	セグメントデータを識別するための ID
SEGSOURCEID	– [uint32]	セグメントソース情報構造体を識別するための ID
nsa_SEGSOURCEINFO	– [struct]	セグメントソース情報構造体

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID または ns_WRONGHEADER
nsObj	– [struct]	Neuroshare データ格納用オブジェクト 【ns_Result が ns_OK である場合】 nsObj.Segment{SEGID}.ns_SEGSOURCEINFO(SEGSOURCEID) 構造体のメンバ変数の各値が更新された nsObj が格納される 【ns_Result が ns_OK でない場合】 入力引数の nsObj がそのまま格納される

ns_AddSegmentData　セグメントデータの追加を行う

~~[ns_Result nsObj SEGSOURCEID] = ns_AddSegmentData(nsObj, SEGID, dTimestamp, dwUnitID, dValue)~~

[ns_Result nsObj] = ns_AddSegmentData(nsObj, SEGID, dTimestamp, dwUnitID, dValue) (※3)

[動作概要]

ns_AddSegmentData()は、~~nsObj,SEGID で指定するセグメントデータに、~~

~~SEGSOURCEID 番目のセグメントソース情報構造体を追加する(※1)~~

セグメントソース情報構造体を追加しない(※3)

そして、dTimestamp, dwUnitID, dValue で表現するデータを追加した(※2)上で、

追加結果(ns_Result) と 追加後の Neuroshare データ格納用オブジェクト(nsObj) ~~と~~

~~追加したセグメントソースヘッダーID(SEGSOURCEID)~~ (※3)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
SEGID	– [uint32]	セグメントデータを識別するための ID
dTimestamp	– [double]	タイムスタンプ値(scalar : 1 × 1)
dwUnitID	– [uint32]	Classification ID 値(scalar : 1 × 1)
dValue	– [double]	セグメント値(vector : 1 × n)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID または ns_WRONGDATA(※4)
nsObj	– [struct]	Neuroshare データ格納用オブジェクト

【ns_Result が ns_OK である場合】

nsObj の各値が更新された nsObj が格納される(※5)

【ns_Result が ns_OK でない場合】

入力引数の nsObj がそのまま格納される

~~SEGSOURCEID – [uint32] セグメントソース情報を識別するための ID~~

~~【ns_Result が ns_OK である場合】~~

~~新規セグメントソース ID 値が格納される~~

~~【ns_Result が ns_OK でない場合】~~

~~''(ブランク)が格納される~~(※3)

[特記事項]

(※1) ~~1 レコード追加とともに、ns_SEGSOURCEINFO が 1 つ追加されます。~~

1 レコード追加が行われた場合においても、ns_SEGSOURCEINFO は追加されません。(※3)

これにより、Entity データのレコード数とセグメントソース構造体の数が一致します。

(※2) Segment Data として、nsObj.Segment[SEGID].FID で指すファイルに

下記順序で Entity データを書き込みます。

このファイルは、ファイル統合(ns_CloseFile)後、自動的に削除されます。

dwSampleCount[scalar : 1 × 1, dValue のデータ数],dTimestamp, dwUnitID, dValue

(※3) Ver1.3 – SegmentEntity の仕様変更(1Segment Entity あたりの SegSourceInfo ヘッダーの登録数制限化)により、本メソッドの機能が以下のように変更されました。

旧

1 ns_SEGSOURCEINFO の個数を増やす(+1)。

2 メソッドの戻り値に SEGSOURCEID を含める。

新

1. ns_SEGSOURCEINFO の個数を増やさない。

2. メソッドの戻り値に SEGSOURCEID を含めない。

Neural Event Entity 登録処理

Neural Event Entity 登録処理に関連するメソッドとその処理概要、使用例等は以下の通りです。

[関連メソッド]

・ns_NewNeuralEventData　・・・　新規ニューラルイベントデータの生成

- ns_TAGELEMENT, ns_ENTITYINFO, ns_NEURALINFO の新規生成、初期化(※1)
- ニューラルイベントデータを識別する ID の新規生成(※2)
- ニューラルイベントデータを格納する中間ファイルの作成
- 構造体の自動更新(※3)

【入力引数 szEntityLabel の内容によっては ERROR 機能 – ns_WRONGLABEL が発生】

・ns_GetNeuralInfo　・・・　ニューラルイベント情報構造体の取得

- nsa_NEURALINFO の取得(※4)

【入力引数 ID の内容によっては ERROR 機能 – ns_WRONGID が発生】

・ns_SetNeuralInfo　・・・　ニューラルイベント情報構造体の更新

- nsa_NEURALINFO の更新(※4)

【入力引数 ID の内容によっては ERROR 機能 – ns_WRONGID が発生】

【入力引数 nsa_NEURALINFO の内容によっては ERROR 機能 – ns_WRONGHEADER が発生】

【入力引数 nsa_NEURALINFO の内容によっては WARNING 機能が発生】

・ns_AddNeuralEventData　・・・　ニューラルイベントデータの追加

- 中間ファイルへの dTimestamp の書き込み
- 構造体の自動更新(※3)

【入力引数 ID の内容によっては ERROR 機能 – ns_WRONGID が発生】

【入力引数 dTimestamp の内容によっては ERROR 機能 – ns_WRONGDATA が発生】

(※1)　初期値は<表 6 nsObj で管理するメンバ変数一覧>を参照してください。

(※2)　この ID 値を元に Get,Set,Add メソッドを使用し、意図した Neural Event Entity に対する操作を行います。

(※3)　更新対象と更新値は<表 2 メソッドが更新するメンバ変数一覧>を参照してください。

(※4)　nsa_NEURALINFO 構造体のメンバ変数は、ns_NEURALINFO のそれと同一です。

[メソッド使用例] ※ >> 以下が Work Space における入力。nsObj 作成後に使用可能。

6. ns_NewNeuralEventData ・・・ 新規イベントデータの生成

>> [retS nsObj NeuralEventID] = ns_NewNeuralEventData(nsObj, 'dummy');

7. ns_GetNeuralInfo ・・・ イベント情報構造体の取得

>> [retT nsa_NEURALINFO] = ns_GetNeuralInfo(nsObj, NeuralEventID);

8. [ユーザーによる構造体の内容の変更]

>> nsa_NEURALINFO.szProbeInfo = 'Words empty of meaning';

9. ns_SetNeuralInfo ・・・ イベント情報構造体の更新

>> [retU nsObj] = ns_SetNeuralInfo(nsObj, NeuralEventID, nsa_NEURALINFO);

10. ns_AddNeuralEventData ・・・ イベントデータの追加

>> [retV nsObj] = ns_AddNeuralEventData(nsObj, NeuralEventID, 1.5);

[注意事項]

・ns_SetNeuralInfo メソッドの使用により、
ns_NEURALINFO 構造体のメンバ変数 dwSourceEntityID/dwSourceUnitID に、
任意の値を設定することが可能です。ただし、正しい値を割り当てるには、他 Entity の個数を把握しなければなりません。詳しくは<Ver. 1.0>を参照してください。

[メソッド-Work Space 関連図]

※自動更新対象は記載省略

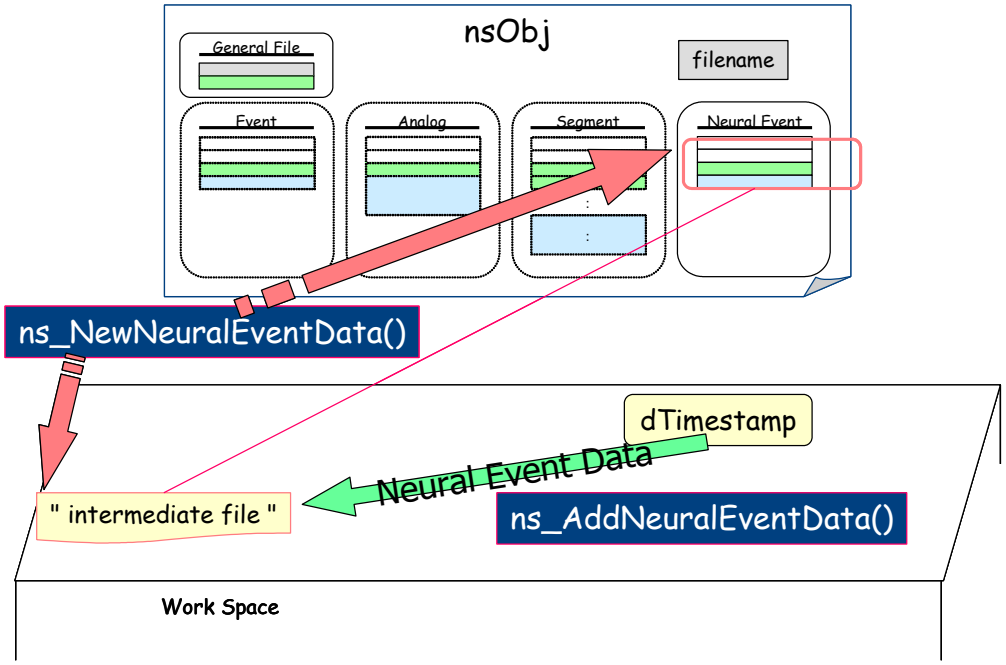


図 14 ns_NewNeuralEventData,ns_AddNeuralEventData

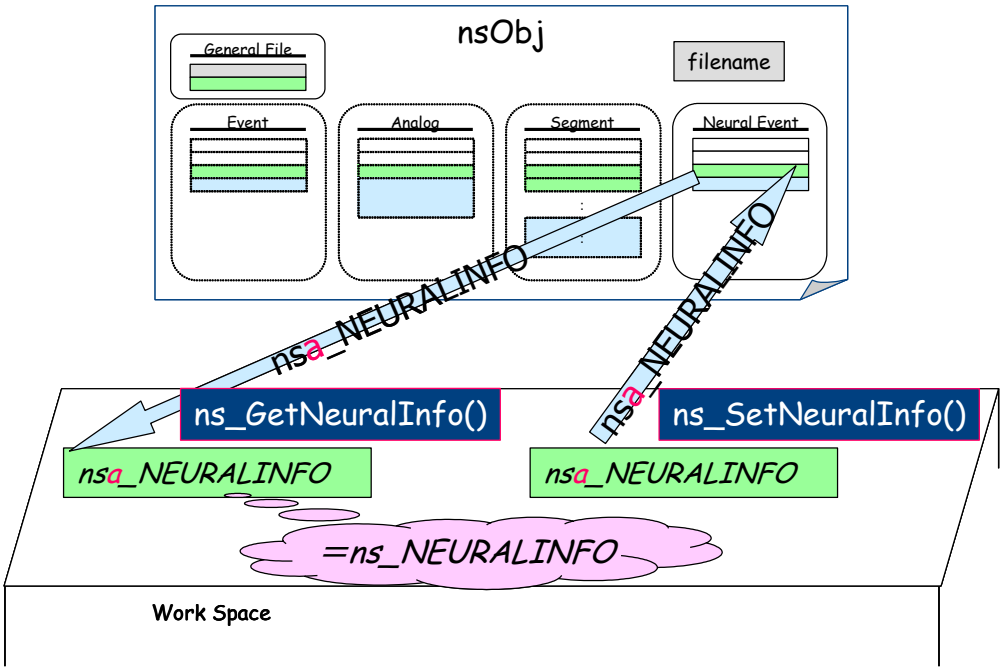


図 15 ns_GetNeuralInfo,ns_SetNeuralInfo

ns_NewNeuralEventData　新規ニューラルイベントデータの生成処理を行う

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

[動作概要]

ns_NewNeuralEventData()は、nsObj に新規ニューラルイベントデータの格納先(中間ファイル)を生成し、
生成結果(ns_Result)，生成処理後の Neuroshare データ格納用オブジェクト(nsObj)，
生成したニューラルイベントデータを識別するための ID を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
szEntityLabel	– [char]	<任意入力項目>新規ニューラルイベントデータの Entity 名(※1)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGLABEL
nsObj	– [struct]	Neuroshare データ格納用オブジェクト 【ns_Result が ns_OK である場合】 nsObj の各値が更新された nsObj(※2)が格納される 【ns_Result が ns_OK でない場合】 入力引数の nsObj がそのまま格納される
ID	– [uint32]	生成したニューラルイベントデータを識別するための ID 【ns_Result が ns_OK である場合】 新規 ID 値が格納される(※3) 【ns_Result が ns_OK でない場合】 ''(ブランク)が格納される

[特記事項]

(※1)　Entity 名の入力は任意です。

Entity 名を省略した(第 2 入力引数が存在しない)場合、これは自動的に''(ブランク)となります。

(※2)　nsObj.NeuralEvent[ID].FID に「作成した中間ファイルへのポインタ」が格納され、

自動更新も実施されています。

尚、ここで作成された中間ファイルは、ファイル統合(ns_CloseFile)後、自動的に削除されます。

(※3)　Neural Event Entity の登録順にみて何番目か、という情報が入ります。

他の Entity 種別の登録数とは無関係です。

ns_GetNeuralInfo　ニューラルイベント情報構造体の取得を行う

[ns_Result nsa_NEURALINFO] = ns_GetNeuralInfo(nsObj, ID)

[動作概要]

ns_GetNeuralInfo()は、nsObj と ID で指定するニューラルイベント情報構造体(nsa_NEURALINFO)を取得し、取得結果(ns_Result) と ニューラルイベント情報構造体を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
ID	– [uint32]	ニューラルイベントデータを識別するための ID

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID
nsa_NEURALINFO	– [struct]	ニューラルイベント情報構造体 【ns_Result が ns_OK である場合】 nsObj,ID で特定される、 nsa_NEURALINFO 構造体が格納される 【ns_Result が ns_OK でない場合】 ''(ブランク)が格納される

ns_SetNeuralInfo　ニューラルイベント情報構造体の更新を行う

[ns_Result nsObj] = ns_SetNeuralInfo(nsObj, ID, nsa_NEURALINFO)

[動作概要]

ns_SetNeuralInfo()は、nsObj と ID で指定するニューラルイベント情報構造体を
nsa_NEURALINFO の内容に置き換え(=更新)、更新結果(ns_Result) と
更新後の Neuroshare データ格納用オブジェクト(nsObj)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
ID	– [uint32]	ニューラルイベントデータを識別するための ID
nsa_NEURALINFO	– [struct]	ニューラルイベント情報構造体

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID または ns_WRONGHEADER
nsObj	– [struct]	Neuroshare データ格納用オブジェクト

【ns_Result が ns_OK である場合】
nsObj.NeuralEvent{ID}.ns_NEURALINFO 構造体の
メンバ変数の各値が更新された nsObj が格納される

【ns_Result が ns_OK でない場合】
入力引数の nsObj がそのまま格納される

ns_AddNeuralEventData　ニューラルイベントデータの追加を行う

[ns_Result nsObj] = ns_AddNeuralEventData(nsObj, ID, dTime)

[動作概要]

ns_AddNeuralEventData()は、nsObj と ID で指定するニューラルイベントデータに、
dTime で表現するデータを追加し（※1）、
追加結果(ns_Result) と 追加後の Neuroshare データ格納用オブジェクト(nsObj)を返す

[入出力引数]

Input :

nsObj	– [struct]	Neuroshare データ格納用オブジェクト
ID	– [uint32]	ニューラルイベントデータを識別するための ID
dTime	– [double]	タイムスタンプ値(scalar : 1×1)

Output :

ns_Result	– [double]	ns_OK または ns_WRONGID または ns_WRONGDATA(※2)
nsObj	– [struct]	Neuroshare データ格納用オブジェクト 【ns_Result が ns_OK である場合】 nsObj の各値が更新された nsObj が格納される(※3) 【ns_Result が ns_OK でない場合】 入力引数の nsObj がそのまま格納される

[特記事項]

(※1)　Neural Event Data として、ns_NewNeuralEventData で作成した nsObj.NeuralEvent{ID}.FID で指す中間ファイルに、下記順序で Entity データを書き込みます。

1. dTime (dTimestamp として)

用語補足

nsObj とは

ns_CreateFile クラスライブラリは、nsObj をクラスライブラリのオブジェクトとしており、
また同時に、Neuroshare 形式のデータを管理する実体としても扱っています。
nsObj をそのように用いることで、Neuroshare 形式のデータ管理を単純化しています。

Neuroshare 形式のデータはいくつかの「構造体+Entity データ」という構成で表現されます。
そこで、ns_CreateFile クラスライブラリは、オブジェクト nsObj を Neuroshare 形式のデータを管理する実体とし、
それに構造体と Entity データの内容を持たせています。(※1)
また、Neuroshare 形式の情報以外にも、変数 filename の情報も併せて持たせています。
filename は ns_CreateFile クラスライブラリの使用によって出力される、
Neuroshare 形式のファイル名を意味しています。<図 16 nsObj の形式>を参照してください。

なお、ns_CreateFile メソッドによって作成された nsObj には、filename の他に、
あらかじめ初期値が格納された sMagicCode,構造体 ns_FILEINFO が含まれます。
各初期値の具体的な値は、ns_CreateFile 以外のメソッドが初期化するメンバ変数も含め、
<表 6 nsObj で管理するメンバ変数一覧>を参照してください。

(※1)　正確には、nsObj は「Entity データの内容が記載されているファイル
(intermediate file, 中間ファイル)へのポインタ」を保持します。

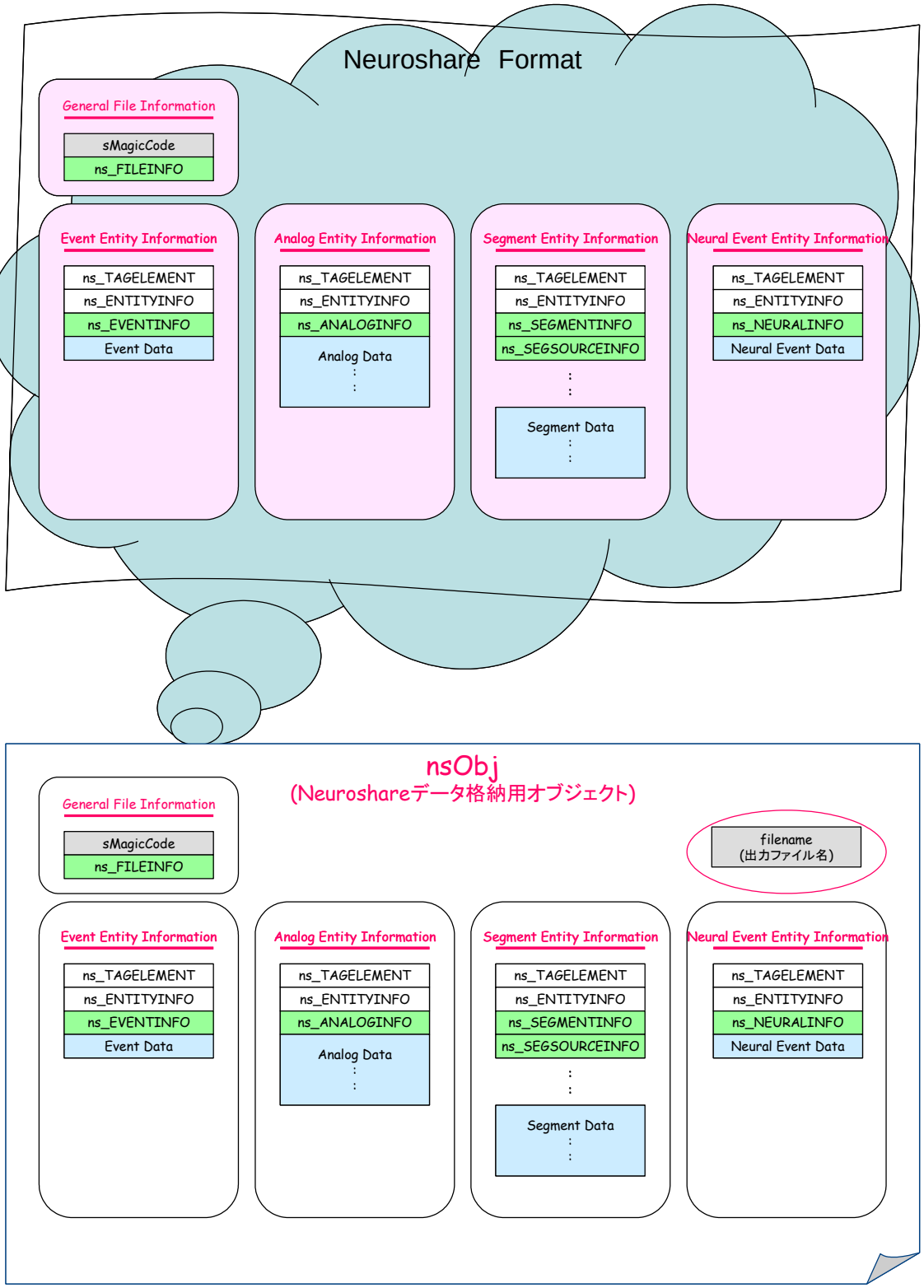


図 16 nsObj の形式

表 6 nsObj で管理するメンバ変数一覧

構造体	(メンバ)変数名	型	初期値	補足 ※空欄は「未設定のため」	Get/Set***Info メソッドによる取得/更新の可/不可 ※「取得メソッドが存在しないための不可」は×印
	sMagicCode	char[16]	'NSN ver000000010'	Document type identifier	×
	filename	char	なし	出力ファイル名。ns_CreateFile 呼び出し時に設定するため初期値なし	×
ns_FILEINFO	szFileType	char[32]	''(ブランク)		可
	dwEntityCount	uint32	0	Entity の数。初期段階で Entity なし	不可(Entity の数に関わるため)
	dTimeStampResolution	double	0		可
	dTimeSpan	double	0		可
	szAppName	char[64]	''(ブランク)		可
	dwTime_Year	uint32	1900	1900 年 1 月 1 日(月) 00:00:00:0000 を初期値としている	可
	dwTime_Month	uint32	1		可
	dwTime_DayOfWeek	uint32	1		可
	dwTime_Day	uint32	1		可
	dwTime_Hour	uint32	0		可
	dwTime_Min	uint32	0		可
	dwTime_Sec	uint32	0		可
	dwTime_MilliSec	uint32	0		可
	szFileComment	char[256]	''(ブランク)		可
ns_TAGELEMENT	dwElemType	uint32	nsObj.CONST.ns_ENTITY_UNKNOWN	Entity の種別。Entity 不明。	×
	dwElemLength	uint32	0		×
ns_ENTITYINFO	szEntityLabel	char[32]	''(ブランク)		×
	dwEntityType	uint32	nsObj.CONST.ns_ENTITY_UNKNOWN	Entity の種別。Entity 不明。	×
	dwItemCount	uint32	0		×
ns_EVENTINFO	dwEventType	uint32	nsObj.CONST.ns_EVENT_TEXT	Event Entity の種別。文字列。	不可(Event Entity のデータ種別のため)
	dwMinDataLength	uint32	0		不可(Event Entity のデータ最小サイズのため)
	dwMaxDataLength	uint32	0		不可(Event Entity のデータ最大サイズのため)
	szCSVDesc	char[128]	''(ブランク)		可
ns_ANALOGINFO	dSampleRate	double	0		可
	dMinVal	double	0		可
	dMaxVal	double	0		可
	szUnits	char[16]	''(ブランク)		可
	dResolution	double	0		可

	dLocationX	double	0		可
	dLocationY	double	0		可
	dLocationZ	double	0		可
	dLocationUser	double	0		可
	dHighFreqCorner	double	0		可
	dwHighFreqOrder	uint32	0		可
	szHighFilterType	char[16]	''(ブランク)		可
	dLowFreqCorner	double	0		可
	dwLowFreqOrder	uint32	0		可
	szLowFilterType	char[16]	''(ブランク)		可
	szProbeInfo	char[128]	''(ブランク)		可
ns_SEGMENTINFO	dwSourceCount	uint32	0		不可(ns_SEGSOURCEINFO 構造体の個数のため)
	dwMinSampleCount	uint32	0		不可(Segment Entity の最小データ数のため)
	dwMaxSampleCount	uint32	0		不可(Segment Entity の最大データ数のため)
	dSampleRate	double	0		可
	szUnits	char[32]	0		可
ns_SEGSOURCEINFO	dMinVal	double	0		可
	dMaxVal	double	0		可
	dResolution	double	0		可
	dSubSampleShift	double	0		可
	dLocationX	double	0		可
	dLocationY	double	0		可
	dLocationZ	double	0		可
	dLocationUser	double	0		可
	dHighFreqCorner	double	0		可
	dwHighFreqOrder	uint32	0		可
	szHighFilterType	char[16]	''(ブランク)		可
	dLowFreqCorner	double	0		可
	dwLowFreqOrder	uint32	0		可
	szLowFilterType	char[16]	''(ブランク)		可
	szProbeInfo	char[128]	''(ブランク)		可
ns_NEURALINFO	dwSourceEntityID	uint32	0		可
	dwSourceUnitID	uint32	0		可
	szProbeInfo	char[128]	''(ブランク)		可

nsa_***INFO 構造体とは

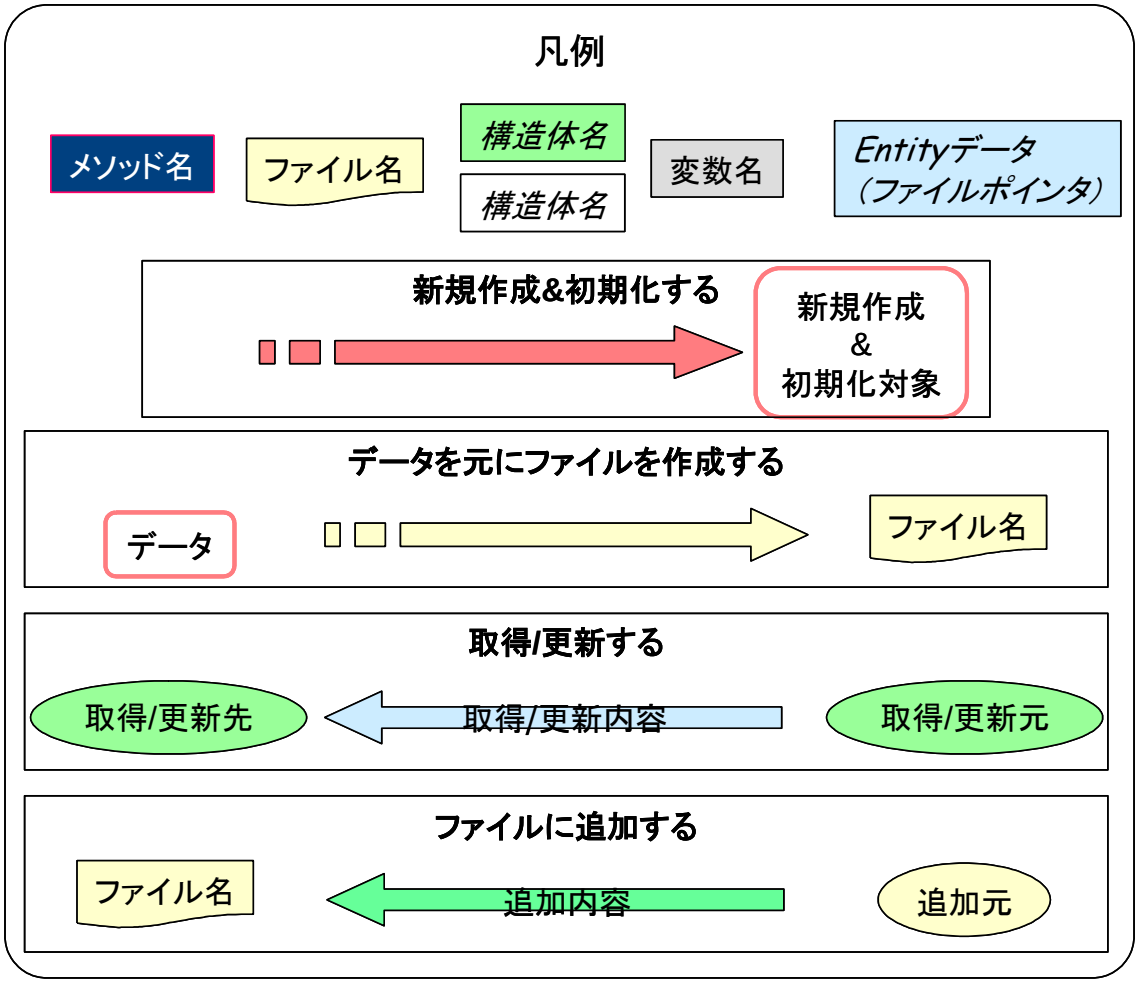
nsa_***INFO 構造体は、
ns_***INFO 構造体から「Entity データとの整合性に関わらないメンバ変数を抽出したもの」であり、
具体的には<表 6 nsObj で管理するメンバ変数一覧>の Get/Set***Info メソッドによる取得/更新の可/不可の項目において「可」となっているメンバ変数の集合を指します。

ns_***INFO 構造体には、
「Entity データとの整合性に関わるメンバ変数」(例 ns_FILEINFO.dwEntityCount [Entity の総数])と、
「Entity データとの整合性に関わらないメンバ変数」(例 ns_FILEINFO.szFileComment [ファイルについてのコメント])が存在します。
このうち、「Entity データとの整合性に関わらないメンバ変数」のみを(※1)、
Get/Set***Info メソッドによって取得/更新が出来るメンバ変数とすることで、
「Entity データとの整合性に関わるメンバ変数」を任意の値に更新できない(データとの整合性を崩さない)ようにしています。

(※1) 例外があります。
下記最大/最小値を意味するメンバ変数に関しては、Entity データの内容によって値が決まりますが、
同時に Get/Set***Info メソッドによる取得/更新が可能です。
(敢えて最大/最小値を任意値に設定したい場合に対応しています。)

ns_ANALOGINFO.dMaxVal
ns_ANALOGINFO.dMinVal
ns_SEGMENTINFO.dMaxVal
ns_SEGMENTINFO.dMaxVal

図表補足


凡例
メソッド名
ファイル名
構造体名
構造体名
変数名
Entityデータ
（ファイルポインタ）
新規作成&初期化する
新規作成
&
初期化対象
データを元にファイルを作成する
データ
ファイル名
取得/更新する
取得/更新先
取得/更新内容
取得/更新元
ファイルに追加する
ファイル名
追加内容
追加元

注意事項

ns_CreateFile クラスライブラリの各種注意点は下記の通りです。

Ver. 1.0

・Entity 種別順でのファイル作成の影響

メソッド ns_CloseFile()は、Neuroshare 形式のファイルを完成させますが、その際、
Event→Analog→Segment→NeuralEvent という Entity 種別順でファイルを作成します。(仕様)

これにより、ns_NEURALINFO.dwSourceEntityID と ns_NEURALINFO.dwSourceUnitID の値を
正しく割り当てるには、全 Entity の種別ごとに計いくつの Entity が存在しているか、
という情報を把握しなければなりません。

・Event Entity 登録処理 CSV 形式未対応

メソッド ns_AddEventData()は、CSV 形式の入力に対応していません。

Ver. 1.3

・1Segment Entity あたりの SegSourceInfo ヘッダーの登録数制限化

SegSourceInfo ヘッダーと SegmentData の関係をより明確にするために、
Ver 1.3 において SegmentEntity の仕様が下記のように変更されました。

旧:

SegmentEntity 数:SegSourceInfo ヘッダー数:SegmentData レコード数 = 1:N:N

新:

SegmentEntity 数:SegSourceInfo ヘッダー数:SegmentData レコード数 = 1:1:N

[SegmentEntity に含まれる SegSourceInfo ヘッダーの数は、1Entity あたり 1 つのみ許可される]

上記仕様変更に伴い、
ns_NewSegmentData()、ns_AddSegmentData()の仕様が変更されました。
変更点の詳細は各メソッドの記載事項を参照してください。

更新履歴

ns_CreateFile クラスライブラリの更新履歴は下記の通りです。

Ver 1.3 Date : 2011/03/04

クラスライブラリの変更(Ver 1.3)に伴う記述修正。

Ver 1.0 Date : 2009/06/25

下記 POMU-Lab site における、ns_CreateFile クラスライブラリの記述を元に設計、実装

POMU-Lab site : http://www2.bpe.es.osaka-u.ac.jp/multineuron/POMU-Lab/index.html