Operation Usage of NeuroShare XOP

Ver 1.0

November 2006

by Takashi Kodama

ns_GetLibraryInfo

This operation returns the information about the library (DLL) included in NeuroShare XOP as global variables. The library is designated in the C source code of NeuroShare.xop.

[Return values]

```
NS_dwLibVersionMaj;              // Major version number of this library.
NS_dwLibVersionMin;              // Minor version number of this library.
NS_dwAPIVersionMaj;              // Major version number of API specification that library complies with
NS_dwAPIVersionMin;              // Minor version number of API specification that library complies with
NS_szDescription[64];           // Text description of the library.
NS_szCreator[64];                // Name of library creator.
NS_dwTime_Year;                  // Year of last modification date
NS_dwTime_Month;                // Month (1-12; January = 1) of last modification date
NS_dwTime_Day;                   // Day of the month (1-31) of last modification date
NS_dwFlags;                      // Additional library flags.
NS_dwMaxFiles                    // Maximum number of files library can simultaneously open.
NS_dwFileDescCount;             // Number of valid description entries in the following array.
NS_FileDesc_szDescription[32]; // Text description of the file type or file family
NS_ FileDesc_szExtension[8];    // Extension used on PC, Linux, and Unix Platforms.
NS_ FileDesc_szMacCodes[8];   // Application and Type Codes used on Mac Platforms.
NS_ FileDesc_szMagicCode[16]; // null-terminated code used at the file beginning.
```

[Memo]

Flags defined:

```
#define ns_LIBRARY_DEBUG 0x01                  // includes debug info linkage
#define ns_LIBRARY_MODIFIED 0x02               // file was patched or modified
#define ns_LIBRARY_PRERELEASE 0x04             // pre-release or beta version
#define ns_LIBRARY_SPECIALBUILD 0x08            // different from release version
#define ns_LIBRARY_MULTITHREADED 0x10           // library is multithread safe
```

ns_GetFilePath

This operation calls OpenFile dialog and get the full path of the file selected as a global variable.

[Return values]

NS_filepath;	// the full path of the file selected by OpenFile dialog

ns_GetFileInfo *file_full_path*

This operation returns the information about the file identified by *file_full_path* as global variables.

[Return values]

NS_ *szFileType[32*];	// Human readable manufacturer's file type descriptor.
NS_ *dwEntityCoun*t;	// Number of entities in the data file. This number is used
	// to enumerate all the entities in the data file from 0 to
	// (dwEntityCount –1) and to identify each entity in
	// function calls (dwEntityID).
NS_ *dTimeStampResolution;*	// Minimum timestamp resolution in seconds.
NS_ *dTimeSpan;*	// Time span covered by the data file in seconds.
NS_ *szAppNam*e[64];	// Information about the application that created the file.
NS_ *dwTime_Year;*	// Year.
NS_ *dwTime_Month;*	// Month (1-12; January = 1).
NS_ dwReserve*d;*	// Used to be - Day of the week (Sunday = 0).
NS_ *dwTime_Day;*	// Day of the month (1-31).
NS_ *dwTime_Hour;*	// Hour since midnight (0-23).
NS_ *dwTime_Min;*	// Minute after the hour (0-59).
NS_ *dwTime_Sec;*	// Seconds after the minute (0-59).
NS_ *dwTime_MilliSec;*	// Milliseconds after the second (0-1000).
NS_ *szFileCommen*t[256];	// Comments embedded in the source file.

ns_GetEntityInfo /E=(*EntityID*) *file_full_path*

This operation returns the information about the Entity identified by *EntityID* and *file_full_path* as global variables.

[Return values]

NS_ *szEntityLabel[32*];	// Specifies the label or name of the entity.
NS_ d*wEntityTyp*e;	// Flag specifying the type of entity data recorded on this
	// channel. It can be one of the following:
	// # define ns_ENTITY_UNKNOWN 0
	// # define ns_ENTITY_EVENT 1
	// # define ns_ENTITY_ANALOG 2
	// # define ns_ENTITY_SEGMENT 3
	// # define ns_ENTITY_NEURALEVENT 4
NS_ *dwItemCoun*t;	// Number of data items for the specified entity in the file.

ns_GetEventInfo /E=(*EntityID*) *file_full_path*

This operation returns the information about the Event Entity identified by *EntityID* and *file_full_path* as global variables.

[Return values]

NS_ *dwEventTyp*e;	// A type code describing the type of event data associated with
	// each indexed entry. The following information types are
	// allowed:
	// #define ns_EVENT_TEXT 0 //text string
	// #define ns_EVENT_CSV 1 //comma separated values
	// #define ns_EVENT_BYTE 2 // 8-bit binary values
	// #define ns_EVENT_WORD 3 //16-bit binary values
	// #define ns_EVENT_DWORD 4 //32-bit binary values
NS_ *dwMinDataLengt*h;	// Minimum number of bytes that can be returned for an Event.
NS_ *dwMaxDataLengt*h;	// Maximum number of bytes that can be returned for an Event.
NS_ *szCSVDesc* [128];	// Provides descriptions of the data fields for CSV Event Entities.

ns_GetEventData /E=(*EntityID*) *file_full_path*

This operation returns the time stamp and data of the Event Entity identified by *EntityID* and *file_full_path* as a wave.

[Return values]

NS_EventTime;	// wave of Event time stamps (unit sec).
NS_EventData;	//wave of Event data.

ns_GetAnalogInfo /E=(*EntityID*) *file_full_path*

This operation returns the information about the Analog Entity identified by *EntityID* and *file_full_path* as global variables.

[Return values]

NS_ *dSampleRate;*	// The sampling rate in Hz used to digitize the analog values.
NS_ *dMinVa*l;	// Minimum possible value of the input signal.
NS_ *dMaxVal;*	// Maximum possible value of the input signal.
NS_ *szUnits[16];*	// Specifies the recording units of measurement.
NS_ *dResolution;*	// Minimum input step size that can be resolved.
	// (E.g. for a +/- 1 Volt 16-bit ADC this value is .0000305).
NS_ *dLocationX;*	// X coordinate of source in meters.
NS_ *dLocationY;*	// Y coordinate of source in meters.
NS_ *dLocationZ;*	// Z coordinate of source in meters.
NS_ *dLocationUser;*	// Additional manufacturer-specific position information
	// (e.g. electrode number in a tetrode).
NS_ *dHighFreqCorner;*	// High frequency cutoff in Hz of the source signal filtering.
NS_ *dwHighFreqOrder;*	// Order of the filter used for high frequency cutoff.
NS_ *szHighFilterType[16];*	// Type of filter used for high frequency cutoff (text format).
NS_ *dLowFreqCorner;*	// Low frequency cutoff in Hz of the source signal filtering.
NS_ *dwLowFreqOrder;*	// Order of the filter used for low frequency cutoff.
NS_ *szLowFilterType[16];*	// Type of filter used for low frequency cutoff (text format)..
NS_ *szProbeInf*o[128*];*	// Additional text information about the signal source.

ns_GetAnalogData /E=(*EntityID*)/R={*start_time*, *time_length*} *file_full_path*

This operation returns the specified length of Analog data as a wave. The time range interested can be specified by *start_time* and *time_length* (unit: sec). The Analog Entity is identified by *EntityID* and *file_full_path*. If time range specification (R={*start_time* ,*time_length*}) is omitted, the whole data is returned.

[Return values]

NS_AnalogData;	//wave of specified length of the Analog data
NS_dwContCount;	*//* Number of continuous data values retrieved.

ns_GetSegmentInfo /E=(*EntityID*) *file_full_path*

This operation returns the information about the Segment Entity identified by *EntityID* and *file_full_path* as global variables.

[Return values]

NS_ *dwSourceCount;*	// Number of sources contributing to the Segment Entity data.
	// For example, with tetrodes, this number would be 4.
NS_ *dwMinSampleCoun*t;	// Minimum number of samples in each Segment data item.
NS_ *dwMaxSampleCoun*t;	// Maximum number of samples in each Segment data item.
NS_ *dSampleRate;*	// The sampling rate in Hz used to digitize source signals.
NS_ *szUnits[32];*	// Specifies the recording units of measurement.

ns_GetSegmentSourceInfo /E=(*EntityID*)/S=(*SourceID*) *file_full_path*

This operation returns the information about the source entity selected by *sourceID,* for the Segment Entity identified by *EntityID* and *file_full_path* as global variables.

[Return values]

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

ns_GetSegmentData /E=(*EntityID*)/I=(*Index*) *file_full_path*

This operation returns the piace of the Segment data specified by *Index,* from the Segment Entity identified by *EntityID* and *file_full_path* ,as a wave and global variables.

[Return values]

NS_SegmentData;	//wave of the Segment data specified by *Index*
NS_SegTime;	//time stamp (unit sec) of the Segment data specified by *Index*
NS_UnitID;	// the unit classification code for the Segment data specified by *Index*
NS_SegSampleCount;	//the number of sampled point in the Segment data specified by *Index*.

ns_GetNeuralInfo /E=(*EntityID*) *file_full_path*

This operation returns the information about the Neural Entity identified by *EntityID* and *file_full_path* as global variables.

[return values]

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

ns_GetNeuralData /E=(*EntityID*)/R={*start_index*, *index_count*} *file_full_path*

This operation returns the time stamp of the Neural data identified by *EntityID* and *file_full_path* as a wave. The index range of interest is specified by *start_index*, *index_count*.

[Return values]

NS_NeuralData;	//wave of time stamps (unit: sec) of the Neural data

ns_GetIndexByTime /E=(*EntityID*)/F=(*Flag*)/T=(*Time*)/Q *file_full_path*

This operation gets the index of the data item which time-stamped at *Time*. The *Flag* specifies whether you want to get the data item that starts before or after the *Time* (unit: sec). If /Q is add in the command, the result is not shown in the history window. The Entity interested is identified by *EntityID* and *file_full_path.*

The *Flag* are defined:

```
#define ns_BEFORE –1          // return the data entry occuring before
                              // and inclusive of the time dTime.
#define ns_CLOSEST 0            // return the data entry occuring at or closest
                              // to the time dTime
#define ns_AFTER +1             // return the data entry occuring after
                              // and inclusive of the time dTime.
```

[Return values]

```
NS_IndexByTime;               //index of the data item which time-stamped at Time
```

ns_GetTimeByIndex /E=(*EntityID*)/I=(*Index*)/Q *file_full_path*

This operation gets the time stamp of the data item which specified by *Index*. If /Q is add in the command, the result is not shown in the history window. The Entity interested is identified by *EntityID* and *file_full_path.*

[Return values]

NS_TImeByIndex;	// the time stamp (unit: sec) of the data item which specified by *Index*