EEG File Format

Project: NEUROWORKS EEG

Date: July 9, 2004

Revision: K

CONFIDENTIAL

1. Introduction

Each EEG study generates four files with the same name, but unique extensions. These four files are known as the patient information, raw data, notes and the table of contents (toc) files with the extensions .eeg, .erd, .ent and .etc respectively.

Each file has two schema numbers which are used to identify version changes to the file format. The information in this document is valid only for the indicated schema numbers.

1.1 Generic File Header : Base Schema 0

Each file begins with a generic file header defined as follows:

Offset (bytes)	Size (bytes)	Name	Type	Description
0	16	m_file_guid	GUID	Globally unique identifier defining the file type. (see below)
16	4	m_file_schema	WORD	Schema number of the file, used to identify version changes.
20	4	m_creation_time	time_t	Creation time in UTC format.
24	4	m_product_version_high	DWORD	NeuroWorks product version – high bytes.
28	4	m_product_version_low	DWORD	NeuroWorks product version - low bytes.
32	80	m_pat_last_name	char [80]	Last name of the patient.
112	80	m_pat_first_name	char [80]	First name of the patient.
192	80	m_pat_middle_name	char [80]	Middle name of the patient.
272	80	m_pat_id	char [80]	ID of the patient.
352				

1.2 Generic File Header : Base Schema 1

Each file begins with a generic file header defined as follows:

Offset (bytes)	Size (bytes)	Name	Type	Description
0	16	m_file_guid	GUID	Globally unique identifier defining the file type. (see below)
16	2	m_file_schema	WORD	Schema number of the file, used to identify version changes.
18	2	m_base_schema	WORD	Schema number of this generic header, common to all headers.
20	4	m_creation_time	time_t	Creation time in UTC format.
24	4	m_patient_id	long	Internal patient ID.
28	4	m_study_id	long	Internal study ID.
32	80	m_pat_last_name	char [80]	Last name of the patient.
112	80	m_pat_first_name	char [80]	First name of the patient.
192	80	m_pat_middle_name	char [80]	Middle name of the patient.
272	80	m_pat_id	char [80]	ID of the patient.
352				

The valid guids are defined as follows:

File Type	GUID (hex)

Patient Information (.eeg)	ca0b0790-d795-11d0-af32-00a0245b54a5
Raw Data (.erd)	ca0b0791-d795-11d0-af32-00a0245b54a5
Table of Contents (.etc)	ca0b0792-d795-11d0-af32-00a0245b54a5
Notes (.ent)	ca0b0793-d795-11d0-af32-00a0245b54a5
Syncronization File (.snc)	6086a9d2-60af-11d3-9860-00104b75c151
Segments Table of Contents (.stc)	e9b71278-a84b-4d59-bf87-10728fb8bd9d

1.3 List Format

Some files encode data in the Excel list format This is a string representation of data in an organized fashion.

A list consists of a series of items. An item can be either another list, or an atomic item. The atomic items contain data in a specific format. The table below outlines the recognized items and their string representations.

List Item	String Representation
Integer **I()**	A signed 4 byte integer number, e.g.: 234 -999
Real **R()**	A signed 8 byte floating point number, e.g.: 3.1415 -4.625E06
String **S()**	A string delimited by quotes. The sequence '\"' is used to indicate a quote within the string. The sequence '\\' is used to indicate a single backslash within the string. E.g.: "Hello" "This character \" is a quote and this \\ is a backslash"
User **U()**	Hex encoded binary data. The first two characters are '0x' followed by 8 hex digits indicating the size of the data. The data immediately follows in hex with two characters per byte. E.g.: 0x0000000401020ab4
List **L()**	A list of items delimited by parentheses and separated by commas. White space characters are ignored. E.g.: (234, 3.1414, "Hello") ("This List", ("Contains", "A List!"))
Key Tree **K(<symbolic name>)**	A hierarchy of name value pairs in list format. A key tree is delimited by parentheses as is a list. The opening parenthesis is followed by the '.' character. A key tree consists of a list of nodes. Each node is a list where the first item is the key name in string format. The second item is optional and corresponds to the data at that location. The data may be any type of item, including another key tree. E.g.: (. (("Weight", 150), ("Height", 68)))
GUID **G()**	Unique identifier. Elements in GUID structure delimited by forward slash '/'. ("GroupId", -441356710/37853/4565/-91/-111/0/1/2/120/104/11)

2. Patient Information File - Schema 3

The patient information file consists of a single null terminated string following the generic file header. The string encodes all the patient information in a list format defined by a hierarchy of name value pairs.

The patient information consists of a single key tree with the symbolic name PatientInfo. The table below indicates the key name, the data type and a description. Compound key names are separated by '.' characters. These refer to sub key trees under the given node.

2.1 K(PatientInfo)

Key	Type	Description
Schema	I()	Schema number of the patient info.
Connections.Sink.Host	S()	Computer name of the storage server.
Connections.Source.Host	S()	Computer name of the signal server.
Info.Name.LastName	S()	
Info.Name.MiddleName	S()	
Info.Name.FirstName	S()	
Info.Address.Address1	S()	
Info.Address.Address2	S()	
Info.Address.City	S()	
Info.Address.State	S()	
Info.Address.Country	S()	
Info.Address.ZIP	S()	
Info.Address.StateLabel	S()	
Info.Address.ZIPLabel	S()	
Info.Admin.Telephone	S()	
Info.Admin.ID	S()	
Info.Admin.BillingID	S()	
Info.Admin.TelephoneLabel	S()	
Info.Admin.IDLabel	S()	
Info.Admin.BillingIDLabel	S()	
Info.Personal.Hand	S()	[“Left” | “Right” | “”]
Info.Personal.Gender	S()	[“Male” | “Female” | “”]
Info.Personal.BirthDate	R()	Microsoft DATE format.
Info.Personal.Height	R()	Units of centimeters.
Info.Personal.Weight	R()	Units of kilograms.
Info.Personal.BirthDateLabel	S()	
Info.Custom.Field1	S()	
Info.Custom.Field2	S()	
Info.Custom.Field3	S()	
Info.Custom.Field4	S()	
Info.Custom.Field5	S()	
Info.Custom.Label1	S()	Title of field1.
Info.Custom.Label2	S()	Title of field2.
Info.Custom.Label3	S()	Title of field3.
Info.Custom.Label4	S()	Title of field4.
Info.Custom.Label5	S()	Title of field5.
Study	L(K(StudyInfo))	List of study information in reverse order of creation time. The head item is the current study.

2.2 K(StudyInfo)

Key	Type	Description
FileName	S()	Full network portable path name.
CreationTime	R()	Microsoft DATE format.
XLTimeZoneInfo	U()	Microsoft TIMEZONEINFO structure recorded with XLCreationTime. Use to convert UTC times to patient locale time.
XLCreationTime	U()	XLTime (currently FILETIME) UTC added in NeuroWorks 2.05

ModificationTime	R()	Microsoft DATE format.
Duration	I()	Milliseconds.
Creator	S()	
Reviewer	S()	Current reviewer.
EegNo	S()	Eeg/Study number
AcquisitionInstrument	S()	Acquisition machine host name.
Headbox	K(Headboxes)	
DSP	K(DspInfo)	
ReviewerInfo	L(K(ReviewerInfo))	List of reviewer information.
Medication	L(K(MedInfo))	List of medication information.
TextField*	K(S(field))	Key tree of user defined text fields in rich text format.
TechField	K(S(field))	Key tree of user defined text fields in rich text format.
PhysField	K(S(field))	Key tree of user defined text fields in rich text format.
Score	K(ScoreChart)	Scoring chart key tree.
ProductVersionHigh	I()	NeuroWorks version - high bytes.
ProductVersionLow	I()	NeuroWorks version - low bytes.
PatientStatus.Normal	I()	BOOL [0 | 1]
PatientStatus.MentallyChallenged	I()	BOOL [0 | 1]
PatientStatus.Awake	I()	BOOL [0 | 1]
PatientStatus.Drowsy	I()	BOOL [0 | 1]
PatientStatus.Asleep	I()	BOOL [0 | 1]
PatientStatus.Uncooperative	I()	BOOL [0 | 1]
PatientStatus.Tense	I()	BOOL [0 | 1]
PatientStatus.Confused	I()	BOOL [0 | 1]
PatientStatus.BehaviorDifficulty	I()	BOOL [0 | 1]
PatientStatus.Aphasic	I()	BOOL [0 | 1]
PatientStatus.SemiComa	I()	BOOL [0 | 1]
PatientStatus.Coma	I()	BOOL [0 | 1]
PatientStatus.StatusEpilepticus	I()	BOOL [0 | 1]
PatientStatus.Other	S()	
LastAttack	R()	Microsoft DATE format.
LastMeal	R()	Microsoft DATE format.
Electrode.Type	I()	[0:disc | 1:collodian | 2:other]
Electrode.OtherType	S()	
Electrode.Placement	I()	[0:10-20 | 1:manual]
Electrode.SpecialLeads	S()	
Electrode.Impedance	S()	
SkullDefects.Background .BitmapHeader	U()	BITMAP structure for the skull picture.
SkullDefects.Foreground .BitmapHeader	U()	BITMAP structure for the defects picture.
SkullDefects.Background.Bitmap	U()	CBitmap::GetBitmapBits()

* No longer defined as of revision C of this document.

SkullDefects.Foreground.Bitmap	U()	CBitmap::GetBitmapBits()

2.3 K(DspInfo)

Key	Type	Description
DSPHWVersion	S()	Hardware version in the format "major_rev.minor_rev"
DSPSWVersion	S()	Software version in the format "major_rev.minor_rev"

2.4 K(Headboxes)

Key	Type	Description
HB0	K(HeadboxInfo)	(optional)
HB1	K(HeadboxInfo)	(optional)
HB2	K(HeadboxInfo)	(optional)
HB3	K(HeadboxInfo)	(optional)

2.5 K(HeadboxInfo)

Key	Type	Description
HBSerialNumber	I()	Headbox serial number.
HBType	I()	Headbox type.
HBSWVersion	S()	Software version in the format "major_rev.minor_rev"
HBIsCalibrated	I()	[TRUE | FALSE]
HBCalDate	I()	UTC format date of last calibration.
HBCalSWVersionHigh	I()	Version of the calibration program. - high bytes.
HBCalSWVersionLow	I()	Version of the calibration program. - low bytes.

2.6 K(ReviewerInfo)

Key	Type	Description
Name	S()	Reviewer's name.
ReviewTime	R()	Time marked as reviewed. Microsoft DATE format.

2.7 K(MedInfo)

Key	Type	Description
Name	S()	Medication name.
Dosage	S()	
StartDate	R()	Microsoft DATE format.
Comment	S()	

2.8 K(ScoreChart)

Key	Type	Description
Entries	L(K(ScoreEntries))	

2.9 K(ScoreEntries)

Key	Type	Description
Time	I()	Time from start of study. [milliseconds]
Title	S()	
Comment	S()	
Duration	I()	[milliseconds]
ShowWaveforms	I()	[TRUE | FALSE]
NoteID	I()	ID of the annotation from which this entry was derived. [> 0]
Montage		K(Montage)

3. Raw Data File

The raw data file (.erd) encodes the sample data in a binary format.

3.1 Raw Data File Header- Schema 5

The raw data file begins with a generic file header followed by a raw data header defined as follows:

Offset (bytes)	Size (bytes)	Name	Type	Description
352	8	m_sample_freq	double	Sample frequency in Hertz.
360	4	m_num_channels	int	Number of channels stored.
364	4	m_deltabits	int	Number of bits of delta information stored per channel. **Currently only a value of 8 is supported.**
368	128	m_phys_chan	int [32]	Storage order to physical channel assignments.
496	16	m_headbox_type	int [4]	Type of the headboxes.
512	16	m_headbox_sn	int [4]	Serial number of the headboxes.
528	40	m_headbox_sw_version	char [4][10]	Headbox software version number: (major_rev.minor_rev).
568	10	m_dsp_hw_version	char [10]	DSP Hardware version number.
578	10	m_dsp_sw_version	char [10]	DSP Software version number.
588	4	m_discardbits	int	Number of least significant bits to discard when storing.
592				

3.2 Raw Data File Header- Schema 6

The raw data file begins with a generic file header followed by a raw data header defined as follows:

Offset (bytes)	Size (bytes)	Name	Type	Description

352	8	m_sample_freq	double	Sample frequency in Hertz.
360	4	m_num_channels	int	Number of channels stored.
364	4	m_deltabits	int	Number of bits of delta information stored per channel. **Currently only a value of 8 is supported.**
368	512	m_phys_chan	int [128]	Storage order to physical channel assignments.
880	16	m_headbox_type	int [4]	Type of the headboxes.
896	16	m_headbox_sn	int [4]	Serial number of the headboxes.
912	40	m_headbox_sw_version	char [4][10]	Headbox software version number: (major_rev.minor_rev).
952	10	m_dsp_hw_version	char [10]	DSP Hardware version number.
962	10	m_dsp_sw_version	char [10]	DSP Software version number.
972	4	m_discardbits	int	Number of least significant bits to discard when storing.
976				

3.3 Raw Data File Header- Schema 7

The raw data file begins with a generic file header followed by a raw data header defined as follows:

Offset (bytes)	Size (bytes)	Name	Type	Description
352	8	m_sample_freq	double	Sample frequency in Hertz.
360	4	m_num_channels	int	Number of channels stored.
364	4	m_deltabits	int	Number of bits of delta information stored per channel. **Currently only a value of 8 is supported.**
368	4096	m_phys_chan	int [1024]	Storage order to physical channel assignments.
4464	16	m_headbox_type	int [4]	Type of the headboxes.
4480	16	m_headbox_sn	int [4]	Serial number of the headboxes.
4496	40	m_headbox_sw_version	char [4][10]	Headbox software version number: (major_rev.minor_rev).
4536	10	m_dsp_hw_version	char [10]	DSP Hardware version number.
4546	10	m_dsp_sw_version	char [10]	DSP Software version number.
4556	4	m_discardbits	int	Number of least significant bits to discard when storing.
4560				

The sample data is encoded using the n-bit delta compression algorithm. For each sample, only the lower n bits of the difference between the last and the current sample are stored. If the difference exceeds the range represented by n bits then a full 32 bit value is stored, thereby ensuring lossless compression.

The raw data file consists of a series of sample packets. A sample packet consists of the sampled value of all channels being stored and an event byte

Each sample packet is stored in three parts:

1. Event Byte

2. Delta Information
3. Absolute Channel Values (if any)

3.3.1 Event Byte

Bit 0 of the event byte indicates the presence of the external trigger during the sample period. At present, a photic stimulator is the only supported device which generates an external trigger.

3.3.2 Delta Information

This is an array of *m_num_channels* of deltas representing the difference between the current and the last channel value. Each element of the array is *n* bits long. The total array is rounded up to an integer number of bytes. If the difference is greater than *max_delta* or less than *min_delta* then the value *abs_delta* is placed in the array.

$$max\_delta = 2^{n-1} - 1$$
$$min\_delta = 1 - 2^{n-1}$$
$$abs\_delta = 2^{n-1}$$

3.3.3 Absolute Channel Values

For each element of the delta array that is set to *abs_delta* a 32 bit integer absolute channel value is stored. If an element of the delta array is not set to *abs_delta* then there is no absolute channel value stored.

Values written to the raw data file consist of undecimated and unfiltered sample data. The conversion factor to engineering units is variable, dependent upon m_headbox_type and m_headbox_sw_version, and which physical channel number is being examined:

m_headbox_type	**m_sw_version**	**Channel Range**	**Conversion Factor (**microvolts per raw sample unit)
1 (EEG32)	All	All	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
3 (EEG128)	All	All	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
6 (EMU36)	All	0-31	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		32-35	$((5000000./(2^{10} - 0.5)) / 2^{6}) * 2^{m\_discardbits}$
4 (AMB28)	All	0-23	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		24-27	$((5000000./(2^{10} - 0.5)) / 2^{6}) * 2^{m\_discardbits}$
9 (MOBEE32)	All	0-32	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		33-34	$(1/2^{6}) * 2^{m\_discardbits}$
8 (MOBEE24)	All	0-24	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		25-26	$(1/2^{6}) * 2^{m\_discardbits}$
5 (HYPPO)	All	0-25	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		26-31	$((8711. / (2^{21} - 0.5)) / (159.8/249.5)) * 2^{m\_discardbits}$
		40-41	$(1/2^{6}) * 2^{m\_discardbits}$
	< 3.4	32-39	$((10000000./(2^{10} - 0.5)) / 2^{6}) * 2^{m\_discardbits}$
	3.4 +	32-39	$(20000000./65536.) * 2^{m\_discardbits}$

3.4 Raw Data File Header- Schema 8

The raw data file begins with a generic file header followed by a raw data header defined as follows:

Offset (bytes)	**Size (bytes)**	**Name**	**Type**	**Description**
352	8	m_sample_freq	double	Sample frequency in Hertz.
360	4	m_num_channels	int	Number of channels stored.

364	4	m_deltabits	int	Number of bits of delta information stored per channel. **Currently only a value of 8 is supported.**
368	4096	m_phys_chan	int [1024]	Storage order to physical channel assignments.
4464	16	m_headbox_type	int [4]	Type of the headboxes.
4480	16	m_headbox_sn	int [4]	Serial number of the headboxes.
4496	40	m_headbox_sw_version	char [4][10]	Headbox software version number: (major_rev.minor_rev).
4536	10	m_dsp_hw_version	char [10]	DSP Hardware version number.
4546	10	m_dsp_sw_version	char [10]	DSP Software version number.
4556	4	m_discardbits	int	Number of least significant bits to discard when storing.
4560	2048	m_shorted	Short [1024]	0's indicate recorded channels, 1's are channels being recorded
6608	2048	m_frequency_factor	Short [1024]	Frequency multipliers: SHRT_MAX indicates headbox frequency, 20 - every 20th sample, 4 - every 4th sample etc.
8656				

M_shorted and m_frequency_factor allow us not to record shorted channels and record channels with sampling frequency lower than the headbox sampling frequency thus reducing file size.

The sample data is encoded using the n-bit delta compression algorithm. For each sample, only the lower n or 2*n bits of the difference between the last and the current sample are stored. If the difference exceeds the range represented by 2*n bits then a full 32 bit value is stored, thereby ensuring lossless compression.

The raw data file consists of a series of sample packets. A sample packet consists of the sampled value of all channels being stored and an event byte

Each sample packet is stored in three parts:

4. Event Byte
5. Frequency byte (Present only if at least one of the m_frequency_factor values in the header is different from SHRT_MAX thus indicating that some channels have been recorded with frequency different from headbox sampling frequency)
6. Delta mask
7. Delta Information
8. Absolute Channel Values (if any)

3.4.1 Event Byte

Bit 0 of the event byte indicates the presence of the external trigger during the sample period. At present, a photic stimulator is the only supported device which generates an external trigger.

3.4.2 Frequency Byte

Optional. This byte is written only if at least one of the m_frequency_factor fields in the header is different from SHRT_MAX, thus indicating that this erd file has channels with recording frequency different from the headbox sampling frequency.
Bit7 – all channels are being writte

Bit5 – Channels with recording frequency of 1/50 of the sampling frequency are written (currently not used)

Bit4 – Channels with recording frequency of 1/20 of the sampling frequency are written
Bit3 – Channels with recording frequency of 1/10 of the sampling frequency are written (currently not used)
Bit2 – Channels with recording frequency of 1/5 of the sampling frequency are written (currently not used)
Bit1 – Channels with recording frequency of 1/4 of the sampling frequency are written
Bit0 – Channels with recording frequency of 1/2 of the sampling frequency are written (currently not used)

Currently variable frequency recording is used for SLEEP studies only.

3.4.3 Delta mask

Bit-mask of a size int(number_of_channels / 8 + 0.5). Each 1 in the mask indicates that corresponding channel has 2*n bit delta, 0 means that corresponding channel has n bit delta.

3.4.4 Delta Information

This is an array of (*m_num_channels – number_of_ shortedchannels – channels_with_different_ recording_frequencies_ that_ are_ not_ recorded_at_ this_ time)* of deltas representing the difference between the current and the last channel value. Each element of the array is *n* or 2*n bits long. The total array is rounded up to int(number_of_channels / 8 + 0.5). If the difference is greater than *max_delta* or less than *min_delta* then the value *abs_delta* is placed in the array.

max_delta = $2^{n*2-1} - 1$
min_delta = $1 - 2^{n*2-1}$
abs_delta = FFFF

3.4.5 Absolute Channel Values

For each element of the delta array that is set to *abs_delta* a 32 bit integer absolute channel value is stored. If an element of the delta array is not set to *abs_delta* then there is no absolute channel value stored.

Values written to the raw data file consist of undecimated and unfiltered sample data. The conversion factor to engineering units is variable, dependent upon m_headbox_type and m_headbox_sw_version, and which physical channel number is being examined:

m_headbox_type	**m_sw_version**	**Channel Range**	**Conversion Factor (**microvolts per raw sample unit)
1 (EEG32)	All	All	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
3 (EEG128)	All	All	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
6 (EMU36)	All	0-31	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		32-35	$((5000000./(2^{10} – 0.5)) / 2^{6}) * 2^{m\_discardbits}$
4 (AMB28)	All	0-23	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		24-27	$((5000000./(2^{10} – 0.5)) / 2^{6}) * 2^{m\_discardbits}$
9 (MOBEE32)	All	0-32	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		33-34	$(1/2^{6}) * 2^{m\_discardbits}$
8 (MOBEE24)	All	0-24	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		25-26	$(1/2^{6}) * 2^{m\_discardbits}$
5 (HYPPO)	All	0-25	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		26-31	$((8711. / (2^{21} - 0.5)) / (159.8/249.5)) * 2^{m\_discardbits}$
		40-41	$(1/2^{6}) * 2^{m\_discardbits}$
	< 3.4	32-39	$((10000000./(2^{10} – 0.5)) / 2^{6}) * 2^{m\_discardbits}$
	3.4 +	32-39	$(20000000./65536.) * 2^{m\_discardbits}$

The *m_phys_chan* array (see Raw Data File Header) indicates the physical channel number of the *m_num_channels* sample values. That is, the first delta value in the delta array corresponds to the physical channel m_phys_chan[0]. The second delta value corresponds to physical channel m_phys_chan[1], etc.

The physical channels are labeled as follows depending on ***m_headbox_type[0]*** in the ERD file header. In case of unknown headbox (not 1,3,4,5,6,8 or 9) a generic label (C***n***) should be used where ***n*** is channel index (1-based).

Channel index	EEG32 Type=1	EEG128 Type=3	EMU36 Type=6	AMB28 Type=4	MOBEE32 Type=9	MOBEE24 Type=8	HYPPO Type=5
0	C3	C1	AC1	AC1	Ref	Ref	C3
1	C4	C2	AC2	AC2	Fp1	Fp1	C4
2	CZ	C3	Ref	Ref	F7	F7	O1
3	F3	C4	Fp1	Fp1	T3	T3	O2
4	F4	C5	F7	F7	A1	A1	A1
5	F7	C6	T3	T3	T5	T5	A2
6	F8	C7	T5	T5	O1	O1	LOC
7	FZ	C8	O1	O1	F3	F3	ROC
8	FP1	C9	F3	F3	C3	C3	CHIN1
9	FP2	C10	C3	C3	P3	P3	CHIN2
10	FPZ	C11	P3	P3	Fpz	Fpz	ECGL
11	O1	C12	Fz	Fz	Fz	Fz	ECGR
12	O2	C13	Cz	Cz	Cz	Cz	LAT1
13	P3	C14	Pz	Pz	Pz	Pz	RAT1
14	P4	C15	F4	F4	Fp2	Fp2	LAT2
15	PZ	C16	C4	C4	F8	F8	RAT2
16	T3	C17	P4	P4	T4	T4	X1
17	T4	C18	Fp2	Fp2	A2	A2	X2
18	T5	C19	F8	F8	T6	T6	X3
19	T6	C20	T4	T4	O2	O2	X4
20	AUX1	C21	T6	T6	F4	F4	X5
21	AUX2	C22	O2	O2	C4	C4	X6
22	AUX3	C23	AC23	AC23	P4	P4	X7
23	AUX4	C24	AC24	AC24	X1	X1	X8
24	AUX5	C25	AC25	DC1	X2	X2	X9
25	AUX6	C26	AC26	DC2	X3		X10
26	AUX7	C27	AC27	DC3	X4		CHEST
27	AUX8	C28	AC28	DC4	X5		ABD
28	PG1	C29	AC29		X6		FLOW
29	PG2	C30	AC30		X7		SNORE
30	A1	C31	AC31		X8		DIF1
31	A2	C32	AC32		X9		DIF2
32		C33	DC1		X10		POSITION
33		C34	DC2				PRES
34		C35	DC3				DC1
35		C36	DC4				DC2

36		C37					DC3
37		C38					DC4
38		C39					DC5
39		C40					DC6
40		C41					OSAT
41		C42					PR
42		C43					
43		C44					
44		C45					
45		C46					
46		C47					
47		C48					
48		C49					
49		C50					
50		C51					
51		C52					
52		C53					
53		C54					
54		C55					
55		C56					
56		C57					
57		C58					
58		C59					
59		C60					
60		C61					
61		C62					
62		C63					
63		C64					
64		C65					
65		C66					
66		C67					
67		C68					
68		C69					
69		C70					
70		C71					
71		C72					
72		C73					
73		C74					
74		C75					
75		C76					
76		C77					
77		C78					
78		C79					
79		C80					
80		C81					

81		C82					
82		C83					
83		C84					
84		C85					
85		C86					
86		C87					
87		C88					
88		C89					
89		C90					
90		C91					
91		C92					
92		C93					
93		C94					
94		C95					
95		C96					
96		C97					
97		C98					
98		C99					
99		C100					
100		C101					
101		C102					
102		C103					
103		C104					
104		C105					
105		C106					
106		C107					
107		C108					
108		C109					
109		C110					
110		C111					
111		C112					
112		C113					
113		C114					
114		C115					
115		C116					
116		C117					
117		C118					
118		C119					
119		C120					
120		C121					
121		C122					
122		C123					
123		C124					
124		C125					
125		C126					

126		C127					
127		C128					

3.5 Raw Data File Header – Schema 9

The raw data file begins with a generic file header followed by a raw data header defined as follows:

Offset (bytes)	Size (bytes)	Name	Type	Description
352	8	m_sample_freq	double	Sample frequency in Hertz.
360	4	m_num_channels	int	Number of channels stored.
364	4	m_deltabits	int	Number of bits of delta information stored per channel. **Currently only a value of 8 is supported.**
368	4096	m_phys_chan	int [1024]	Storage order to physical channel assignments.
4464	16	m_headbox_type	int [4]	Type of the headboxes.
4480	16	m_headbox_sn	int [4]	Serial number of the headboxes.
4496	40	m_headbox_sw_version	char [4][10]	Headbox software version number: (major_rev.minor_rev).
4536	10	m_dsp_hw_version	char [10]	DSP Hardware version number.
4546	10	m_dsp_sw_version	char [10]	DSP Software version number.
4556	4	m_discardbits	int	Number of least significant bits to discard when storing.
4560	2048	m_shorted	Short [1024]	0's indicate recorded channels, 1's are channels being recorded
6608	2048	m_frequency_factor	Short [1024]	Frequency multipliers: SHRT_MAX indicates headbox frequency, 20 - every 20th sample, 4 - every 4th sample etc.
8656				

M_shorted and m_frequency_factor allow us not to record shorted channels and record channels with sampling frequency lower than the headbox sampling frequency thus reducing file size.

The sample data is encoded using the n-bit delta compression algorithm. For each sample, only the lower n or 2*n bits of the difference between the last and the current sample are stored. If the difference exceeds the range represented by 2*n bits then a full 32 bit value is stored, thereby ensuring lossless compression.

The raw data file consists of a series of sample packets. A sample packet consists of the sampled value of all channels being stored and an event byte

Each sample packet is stored in three parts:

9. Event Byte
10. Frequency byte (Present only if at least one of the m_frequency_factor values in the header is different from SHRT_MAX thus indicating that some channels have been recorded with frequency different from headbox sampling frequency)
11. Delta mask
12. Delta Information
13. Absolute Channel Values (if any)

3.5.1 Event Byte

Bit 0 of the event byte indicates the presence of the external trigger during the sample period. At present, a photic stimulator is the only supported device which generates an external trigger.

3.5.2 Frequency Byte

Optional. This byte is written only if at least one of the m_frequency_factor fields in the header is different from SHRT_MAX, thus indicating that this erd file has channels with recording frequency different from the headbox sampling frequency.
Bit7 – all channels are being writte

Bit5 – Channels with recording frequency of 1/50 of the sampling frequency are written (currently not used)
Bit4 – Channels with recording frequency of 1/20 of the sampling frequency are written
Bit3 – Channels with recording frequency of 1/10 of the sampling frequency are written (currently not used)
Bit2 – Channels with recording frequency of 1/5 of the sampling frequency are written (currently not used)
Bit1 – Channels with recording frequency of 1/4 of the sampling frequency are written
Bit0 – Channels with recording frequency of 1/2 of the sampling frequency are written (currently not used)

Currently variable frequency recording is used for SLEEP studies only.

3.5.3 Delta mask

Bit-mask of a size int(number_of_channels / 8 + 0.5). Each 1 in the mask indicates that corresponding channel has 2*n bit delta, 0 means that corresponding channel has n bit delta.

3.5.4 Delta Information

This is an array of (*m_num_channels – number_of_ shortedchannels – channels_with_different_ recording_frequencies_ that_ are_ not_ recorded_at_ this_ time)* of deltas representing the difference between the current and the last channel value. Each element of the array is *n* or 2*n bits long. The total array is rounded up to int(number_of_channels / 8 + 0.5). If the difference is greater than *max_delta* or less than *min_delta* then the value *abs_delta* is placed in the array.

max_delta = $2^{n*2-1} - 1$
min_delta = $1 - 2^{n*2-1}$
abs_delta = FFFF

3.5.5 Absolute Channel Values

For each element of the delta array that is set to *abs_delta* a 32 bit integer absolute channel value is stored. If an element of the delta array is not set to *abs_delta* then there is no absolute channel value stored.

Values written to the raw data file consist of undecimated and unfiltered sample data. The conversion factor to engineering units is variable, dependent upon m_headbox_type and m_headbox_sw_version, and which physical channel number is being examined:

m_headbox_type	**m_sw_version**	**Channel Range**	**Conversion Factor (**microvolts per raw sample unit)
1 (EEG32)	All	All	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
3 (EEG128)	All	All	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
6 (EMU36)	All	0-31	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		32-35	$((5000000./(2^{10} – 0.5)) / 2^{6}) * 2^{m\_discardbits}$
4 (AMB28)	All	0-23	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$

		24-27	$((5000000./(2^{10} - 0.5)) / 2^6) * 2^{m\_discardbits}$
9 (MOBEE32)	All	0-32	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		33-34	$(1 / 2^6) * 2^{m\_discardbits}$
8 (MOBEE24)	All	0-24	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		25-26	$(1 / 2^6) * 2^{m\_discardbits}$
5 (HYPPO)	All	0-25	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		26-31	$((8711. / (2^{21} - 0.5)) / (159.8/249.5)) * 2^{m\_discardbits}$
		40-41	$(1 / 2^6) * 2^{m\_discardbits}$
	< 3.4	32-39	$((10000000./(2^{10} - 0.5)) / 2^6) * 2^{m\_discardbits}$
	3.4 +	32-39	$((20000000./65536.) / 2^6) * 2^{m\_discardbits}$
14 (Connex)	All	0-37	$(8711. / (2^{21} - 0.5)) * 2^{m\_discardbits}$
		38-47	$((10800000./65536.) / 2^6) * 2^{m\_discardbits}$
		48-49	$(1 / 2^6) * 2^{m\_discardbits}$
15 (Trex)	All	0-23	$(10710./(2^{21} - 0.5)) * 2^{m\_discardbits}$
		24-27	$((10710./(2^{21} - 0.5)) / -1.163) * 2^{m\_discardbits}$
		28-31	$((10000000. / 65536.) / 2^6) * 2^{m\_discardbits}$
		32-33	$(1 / 2^6) * 2^{m\_discardbits}$

The *m_phys_chan* array (see Raw Data File Header) indicates the physical channel number of the *m_num_channels* sample values. That is, the first delta value in the delta array corresponds to the physical channel m_phys_chan[0]. The second delta value corresponds to physical channel m_phys_chan[1], etc.

The physical channels are labeled as in schema 8, with the addition of the two new headboxes listed below, depending on ***m_headbox_type[0]*** in the ERD file header. In case of unknown headbox (not 1,3,4,5,6,8,9,14 or 15) a generic label (C***n***) should be used where ***n*** is channel index (1-based).

Channel index	Connex EEG/Sleep Type=14	Trex Ambulatory Type=15
0	C3	Fp1
1	C4	F7
2	O1	T3
3	O2	A1
4	A1	T5
5	A2	O1
6	Cz	F3
7	F3	C3
8	F4	P3
9	F7	Fpz
10	F8	Fz
11	Fz	Cz
12	Fp1	Pz
13	Fp2	Fp2
14	Fpz	F8
15	P3	T4
16	P4	A2

17	Pz	T6
18	T3	O2
19	T4	F4
20	T5	C4
21	T6	P4
22	LOC	X1
23	ROC	X2
24	CHIN1	DIF1
25	CHIN2	DIF2
26	ECGL	DIF3
27	ECGR	DIF4
28	LAT1	DC1
29	LAT2	DC2
30	RAT1	DC3
31	RAT2	DC4
32	CHEST	OSAT
33	ABD	PR
34	FLOW	
35	SNORE	
36	DIF5	
37	DIF6	
38	POS	
39	DC2	
40	DC3	
41	DC4	
42	DC5	
43	DC6	
44	DC7	
45	DC8	
46	DC9	
47	DC10	
48	OSAT	
49	PR	

4. Note File - Schema 3

The note file contains notes entered during acquisition and review. Some notes are automatically generated by the NeuroWorks software, and some are entered directly by the user.

The note file begins with a generic EEG file header. After this is a series of variable length records. Each record begins with a fixed length header defined as follows:

Offset (bytes)	Size (bytes)	Name	Type	Description
0	4	Type	int	One of: END_OF_NOTES_TYPE = 0, KEYTREE_TYPE = 1

4	4	Length	long	length of the record including the header in bytes.
8	4	prev_length	long	length of the last record including the header.
12	4	Id	long	**This field is unused. The unique ID is now stored under the "GUID" key in the note key tree.**

The last record in the file consists only of a note header with type of 0 (END_OF_NOTES_TYPE). All other records have type 1 (KEYTREE_TYPE). For key tree records, the data which follows the header is in Excel list format. This is a string representation of a key tree with the following format:

4.1 K(Note)

Key	Type	Description
Type	S()	"Annotation"
GUID	S()	Unique note id number within the study.
Text	S()	Note title.
Comment	S()	Note comment.
Stamp	I()	Sample stamp of the note. This is a count of samples from the start of the study indicating the time of the note.
Origin	S()	"Review" or "Acquisition"
Host	S()	The name of the computer on which the note was generated.
ReadOnly	I()	TRUE or FALSE
Data	K(NoteData)	Additional note data.

4.2 K(NoteData)

4.2.1 Base Note Data

All notes have the following note data fields:

Key	Type	Description
Type	S()	"", "Montage", "Montage Delta", "Impedance", "Reference"
User	S()	User login name of the note creator.
CreationTime	R()	Microsoft DATE format.
ModificationTime	R()	Microsoft DATE format.
Deleted	I()	TRUE or FALSE. May not be present if note is not deleted.

Depending on the note data type additional fields are present, as outlined below.

4.2.2 Impedance Note Data

Key	Type	Description
ThresholdIndex	I()	25000 Kohm = 0, 10000 Kohm = 1, 5000 Kohm = 2,

		2500 Kohn = 3
IsDoneScanning	I()	TRUE or FALSE
CurrentChannel	I()	
LockedChannel	I()	
Chan*x*	I()	Impedance value for channel number *x* in ohms.
SD*x*	R()	Standard deviation of the impedance.

4.2.3 Montage Note Data

Key	Type	Description
FileName	S()	Full path name of the montage file.
Schema	I()	2
Name	S()	Montage name.
MontgageType	S()	"10-20", "Grid"
HalfMontageChannels	L(K(HalfMontageChannel))	
Channels	L(K(ChanNode))	
ChanNames	L(S())	List of channel names.
SpecResolution	I()	Spectrum resolution.
SpecDecimation	I()	Spectrum decimation.
SpecNumBins	I()	Number of spectral bins.
Bin0 ... Bin5	K(SpecBinData)	

4.2.3.1 K(SpecBinData)

Key	Type	Description
LowFreq	R()	Lower bin frequency [Hz].
HighFreq	R()	Higher bin frequency [Hz].
Color	I()	Bin color. (COLORREF)

4.2.3.2 K(ChanNode)

Key	Type	Description
ChanIndex	I()	Output channel index.
Channel	K(MontageChannel)	

4.2.3.3 K(MontageChannel)

Key	Type	Description
ChanType	S()	"Waveform", "Amplitude Spectrum [µV]", "Power Spectrum [uV²]" "Power Spectrum [dB]"
Group	I()	
Page	I()	
Color	I()	COLORREF
Gain	S()	Gain string (number followed by a unit)
LffCutoff	R()	[Hz]
HffCutoff	R()	[Hz]

NotchCutoff	R()	[Hz]
From_Name	S()	Input2 half montage channel name.
To_Name	S()	Input1 half montage channel name
ChanProcType	S()	Used to indicate if event detection is active for this channel. Possible values: “Enabled”, (used for detection) “Disabled”, (not used) “Artifact” (used to detect eye blinks, etc.)
PositiveUp	I()	0 / 1
GroupId	G()	
Set	I()	
IChannelId	G()	
ITypeId	G()	
ISiteId	G()	
IInputId	G()	
SamplingFreq	R()	
Calibration	K()	
OChannelId	G()	
OTypeId	G()	

4.2.3.4 K(HalfMontageChannel)

Key	Type	Description
Name	S()	
Readonly	I()	TRUE or FALSE. HMC’s that correspond to physical channels are read only. The Laplacian HMC’s are writeable.
Coefficients	L(K(Coefficients))	

4.2.3.5 K(Coefficients)

Key	Type	Description
Chan	I()	Input channel Id.
Coeff	R()	Channel weight.

4.2.4 Montage Delta Note Data

Key	Type	Description
0 … N	K(ChannelDelta)	Change in channel N.

4.2.4.1 K(ChannelData)

Key	Type	Description
Gain	R()	Gain in uV/mm. Only present if changed.
LffCutoff	R()	[Hz] . Only present if changed.
HffCutoff	R()	[Hz] . Only present if changed.
NotchCutoff	R()	[Hz] . Only present if changed.

4.2.5 Event and Episode, or Spike and Seizure Note

Key	Type	Description
CreationTime	R()	Microsoft DATE format
UseCreator	I()	1, if created by a processor. 0 if manually creator
ChannelNumber	I()	The channel number of the event/episode if applicable.
StartTime	_int64 (FileTime)	The start time of the note saved in FileTime format.
EndTime	_int64 (FileTime)	The end time of the note, if applicable, saved in FileTime format.
EndStamp	I()	End stamp of the note, if applicable, -1 otherwise
AnalysisId	U()	GUID that identifies which batch analysis run the note belongs to. Only present for notes generated by batch analyzer.
AnalysisContext	I()	Indicates which application created the note. Set to one of: 1 – Batch Analyzer 2 – Ambulatory Upload 3 – Wave, Acquisition 4 – Wave, Review At the time of writing only the batch analyzer creates and sets this key. It is used by the batch analyzer to distinguish between notes created by it and the other applications.

5. Table Of Contents File

The Table Of Contents file is an index into the raw data file used for mapping samples to times, and also to facilitate quick searches. This file ends in the extension '.toc'. The file starts with a generic EEG file header, and is followed by a series of fixed length records called toc entries.

Each toc entry provides an offset into the raw data file to the start of a sample packet. The packet at that location is guaranteed to contain all abs_delta values in the delta information, followed by absolute sample values.

A toc entry is added to the toc file whenever recording is started and also for every 1000 samples.

Note: The byte alignment for the toc entry structure is set at 4 bytes.

5.1 TOC Entry – File Schema 2

Offset (bytes)	Size (bytes)	Name	Type	Description
0	4	offset	DWORD	Offset in bytes into the '.erd' file to a sample

				packet.
4	4	timestamp	long	Timestamp in milliseconds from the start of the study. The timestamp keeps incrementing even when not recording.
8	4	sample_num	long	Number of samples recorded to this point.
12	4	sample_span	short	Number of samples from this toc entry to the next.

5.2 TOC Entry – File Schema 3

Offset (bytes)	Size (bytes)	Name	Type	Description
0	4	offset	DWORD	Offset in bytes into the '.erd' file to a sample packet.
4	4	samplestamp	long	Sample count from the start of the study. Once the study begins, the sample count keeps incrementing at the sampling frequency, even when not recording. The sampling frequency is constant throughout a study, so the sample stamp can be readily converted to time.
8	4	sample_num	long	Number of samples recorded to this point.
12	4	sample_span	short	Number of samples from this toc entry to the next.

6. Synchronization File (.SNC)

The synchronization file is used to calculate a FILETIME given a sample stamp (and vise-versa). Theoretically, it is possible to calculate a sample stamp's FILETIME given the FILETIME of sample stamp zero (when sampling started) and the sample rate. However, because the sample rate cannot be represented with full precision the accuracy of the FILETIME calculation is affected.

To compensate for the lack of accuracy, the synchronization file maintains a sample stamp-to-computer time (called, MasterTime) mapping. Interpolation is then used to calculate a FILETIME given a sample stamp (and vise-versa).

6.1 SNC file entry – Schema 0

The synchronization file begins with a generic file header followed by 1 or more synchronization entries:

Size (bytes)	Name	Type	Description
4	sampleStamp	long	Sample number from start of study.
8	sampleTime	FILETIME	File time representation of sampleStamp.

The attributes, sampleStamp and sampleTime, are used to predict (using interpolation) the FILETIME based upon a given sample stamp (and vise-versa). Currently, the only use for this conversion process is to enable correlation of EEG (sample_stamp) data with other sources of data such as Video (which works in FILETIME).

7. Segment Table of Contents File (.STC)

The Segment Table of Contents file is an index into pairs of (raw data file / table of contents file). It is used for mapping samples file segments. EEG raw data is split into segments in order to break a single file size limit (used to be 2GB) while still allowing quick searches. This file ends in the extension '**.stc**'. Default segment size (size of ERD file after which it is closed and new [ERD / ETC] pair is opened) is 50MB and may be tweaked by **[HKCU]\Software\ExcelTech\Storage\Settings\SegmentSize<Low | High>** registry settings. The file starts with a generic EEG file header, and is followed by a series of fixed length records called the STC entries. ERD segments are named according to the following schema:
<FIRST_NAME>, <LAST_NAME>_<GUID>.ERD (first)
<FIRST_NAME>, <LAST_NAME>_<GUID>.ETC (first)
<FIRST_NAME>, <LAST_NAME>_<GUID>_<INDEX>.ERD (second and subsequent files)
<FIRST_NAME>, <LAST_NAME>_<GUID>_<INDEX>.ETC (second and subsequent files)

<INDEX> is formatted with "%03d" format specifier and starts at 1 (initial value being 0 and omitted for compatibility with the previous versions).

Each STC entry provides an index of the ERD / ETC pair for specific range of samples.
An STC entry is added to the STC file whenever a new segment is created.

Note: The byte alignment for the STC entry structure is set at 4 bytes.

7.1 STC File Header- Schema 1

The stc data file begins with a generic file header followed by a stc data header defined as follows:

Offset (bytes)	Size (bytes)	Name	Type	Description
352	4	m_next_segment	int	Sample frequency in Hertz.
356	4	m_final	int	Number of channels stored.
360	48	m_padding	int[12]	Padding
402				

7.2 STC Entry – File Schema 1

STC entry is very similar to the ETC file entry so that current algorithms that rely on last ETC entry to get end of study stamp can still work based on information in STC file.

Offset (bytes)	Size (bytes)	Name	Type	Description
0	4	segment_name	char[256]	Name of ERD / ETC file segment.
256	4	start_stamp	long	First sample stamp that is found in the ERD / ETC pair.
260	4	end_stamp	long	Last sample stamp that is still found in the ERD / ETC pair.
264	4	sample_num	long	Number of samples recorded to the point that corresponds to **start_stamp**. This number accumulates over ERD / ETC pairs and is equal to **sample_num** of the first entry in the ETC file referenced by this STC entry.
12	4	sample_span	long	Number of samples from this stc entry to the next.

8. Sleep Data File (.EPO)

The Sleep Data file stores all sleep specific data. It stores individual data items related to a study (such as epoch length, all fields will be discussed further below), as well as channels storing data for each epoch. The file contains a header, which has all of the individual data items, and a group of individual data channels, each containing as many data items as the number of epochs in the whole study (calculated from start of recording to end of recording, independent of the lights off time). The sleep data file uses structured storage to store its data, which is similar to a file system, with "Storages" representing directories, and "streams" representing files. It uses the following format:

<root> (storage)
- **<GlobalHeader> (storage)**
  - **<Schema> (stream)**
  - **<EpochInfo> (stream)**
  - **<StudyInfo> (stream)**
- **<Epoch Data Channel> (storage)**
  - **<ChannelInfo> (stream)**
  - **<ChannelRawData> (stream)**
- **<Epoch Data Channel> (storage)**
- **<Epoch Data Channel> (storage)**
- **…**

8.1 EPO File Header (Schema 3.4.0.0)

The epo file header stores individual data items, separated into 3 separate streams: Schema (which stores the file schema), StudyInfo (which stores the patient and study guid associated with this file), and epoch info (which stores all other sleep related individual data items). These data items are always written/read from the file in the exact same order, as seen in the EpochDataBridge project.

8.1.1 EPO File Header Schema

Name	Type	Size (bytes)	Description
Major	Int	4	Major version number
Minor	Int	4	Minor version number
Revision	Int	4	Revision number
Build	int	4	Build number

8.1.2 EPO File Header StudyInfo

Name	Type	Size (bytes)	Description
StudyGuid	GUID	9	Study guid associated with this file
PatientGuid	GUID	9	Patient guid associated with this file

8.1.3 EPO File Header EpochInfo

Name	Type	Size (bytes)	Description
EpochLength	Long	4	Epoch length for the study (in seconds)
TotalEpochs	Long	4	Number of epochs in the study (from lights off to end of study)
CurrentEpoch	Long	4	The current epoch being looked at while reviewing the study

StartTime	XLTIME_T	4	The sample stamp corresponding to the start of epoch #1 in the study
SleepTimeBase	Long	4	The timebase used for sleep studies (obsolete)
Valid	BYTE	1	The valid flag to determine whether or not all data in the file is complete.
CreationTime	XLTIME_T	4	Unused
CurrentStageName	BSTR	Variable	The "friendly name" of the current staging set being written to/read from.
SegmentStartStamp	Long	4	Unused (originally intended to be used with merging and auto-recovery)
StudyStartStamp	Long	4	The sample stamp corresponding to the start of the study
EpochOffset	Long	4	The number of epochs in the study before lights off
RecoverEpochOffset	Long	4	Unused (another variable intended for use with merging and auto-recovery)

8.2 EPO File Epoch Data Channel (Schema 3.4.0.0)

The storages corresponding to Epoch Data Channels have GUIDS for names to ensure that there are no attempts to create storages with the same name (since channel friendly names could theoretically be the same). Thus, each storage has 2 streams within it, one to store channel information and one to store the actual data for the channel.

8.2.1 EPO File Epoch Data Channel Basic ChannelInfo Stream

The following fields are found in the ChannelInfo stream for ALL epoch data channels

Name	**Type**	**Size (bytes)**	**Description**
FriendlyName	BSTR	Variable	The human readable channel name
ChannelType	enChannelType	4	The type of channel (Header, Staging Set, Marking Set, Event Summary, or Data Summary)
StartEpoch	Long	4	Unused
EndEpoch	Long	4	Unused
Hidden	VARIANT_BOOL	2	Determines whether or not the channel is hidden
Creator	BSTR	Variable	Unused (but would appear that it is supposed to be the name of the person who created the channel)
Password	BSTR	Variable	Password used for protecting channels from unwanted changes
Description	BSTR	Variable	Unused (intended as a text description of the channel)

8.2.2 EPO File Epoch Data Channel DataSummary ChannelInfo Stream

The following fields are found in the ChannelInfo stream for DataSummary channels, in addition to the basic fields found in section 8.2.1

Name	**Type**	**Size (bytes)**	**Description**
DataType	enDataType	4	The type of data found in this channel (EKG, O2SAT, HeartRate, Body Position or CPAP)
SummaryType	enSummaryType	4	The type of summary data (count, duration, average, min, max, position, Inspiratory CPAP, Expiratory CPAP, or Breath Rate)

8.2.3 EPO File Epoch Data Channel EventSummary ChannelInfo Stream

The following fields are found in the ChannelInfo stream for EventSummary channels, in addition to the basic fields found in section 8.2.1. This channel type is specifically used for storing sleep event information.

Name	Type	Size (bytes)	Description
Represents	enEventRepType	4	The type of event data that this represents (event count, duration, maximum amplitude, average amplitude, or minimum amplitude)
TypeID	BSTR	Variable	The type ID of the sleep event
EventTypeDesc	BSTR	Variable	Unused (intended as a text description of the event type)

8.2.4 EPO File Epoch Data Channel RawData Stream

The RawData stream in the Epoch Data Channel is a continuous byte stream (of size (TotalEpochs + EpochOffset) * sizeof(datatype) according to the header) which can be interpreted as a stream of varying types, depending on the type of channel. The following table explains how to interpret the data based on the channel type.

ChannelType	Type of Data in Stream	Size
Header	See header description	N/A
Staging Set	Byte	1
Marking Set	Byte	1
Data Summary	Double	8
Event Summary	Double	8

9. Video Table of Contents File (.VTC)

VTC files may have two possible extensions:

Ordinary VTC file - *.vtc
Extracted video files - *.new.vtc[1]

VTC file consists of a header and zero or more records. Structure of a VTC file can be expressed as:

VtcFile := VtcHeader { VtcRecord }

```
VtcHeader :=    GUID_VTC,              // 16 bytes
                LONG_Schema            // 4 bytes

VtcRecord :=    STRING_MpgFileName,    // 264 bytes (261 aligned to 4-byte boundary)
                INT_View,              // 4 bytes
                FILETIME_StartTime,    // 8 bytes
                FILETIME_EndTime       // 8 bytes
```

Where

GUID_VTC = {5E01C1F1-F9A9-11D2-ADF3-00104B8CC65E}
LONG_Schema = 1

[1] It is an additional *.vtc file that has links to an optional set of mpg files. This set is created when user uses "Extract Video" in pruning clips dialog. It is represented by a red line beneath a blue one that represents the original video.

STRING_MpgFileName is a 261 bytes character array
INT_View = 0
FILETIME_StartTime and FILETIME_EndTime are time of the first and the last video frame stored in MPG file referred by the VtcRecord.