The NorPix Sequence File Format (.seq)

Sequence files

Overview

The sequence files created by StreamPix/TroublePix/Hermes while recording (*.seq* extension) use the NorPix Sequence File Format explained here.

A sequence file is made of a header section located in the first 1024 bytes. The header contains information pertaining to the whole sequence: image size and format, frame rate, number of images etc.

Following the header, each images is stored and aligned on a 8192 bytes boundary. Please note that only the uncompressed sequence format is documented here, compressed sequences are handled in a different way.

Usually, pixels in the images are stored for top-left to bottom-right corner. Immediately following the image data comes 8 bytes, containing the absolute timestamp at which the image has been grabbed. The first 4 bytes hold the date and time, then 2 bytes for the milliseconds and the last 2 bytes are the microseconds.

Example

Here is an example for a sequence of 10 images of 640 x 480 pixels in 8 bit monochrome in which the first image in the sequence file is at an offset of 8192 bytes (no metadata).

Read 640 x 480 or 307200 bytes to get all the image pixels. Then read the next 32 bit (4 bytes) to get the timestamp in seconds, formatted according to the C standard *time_t* data structure (32-bit). Read the next 16 bit (2 bytes) as an unsigned short to get the millisecond precision on the timestamp, then read the last 2 for the microseconds.

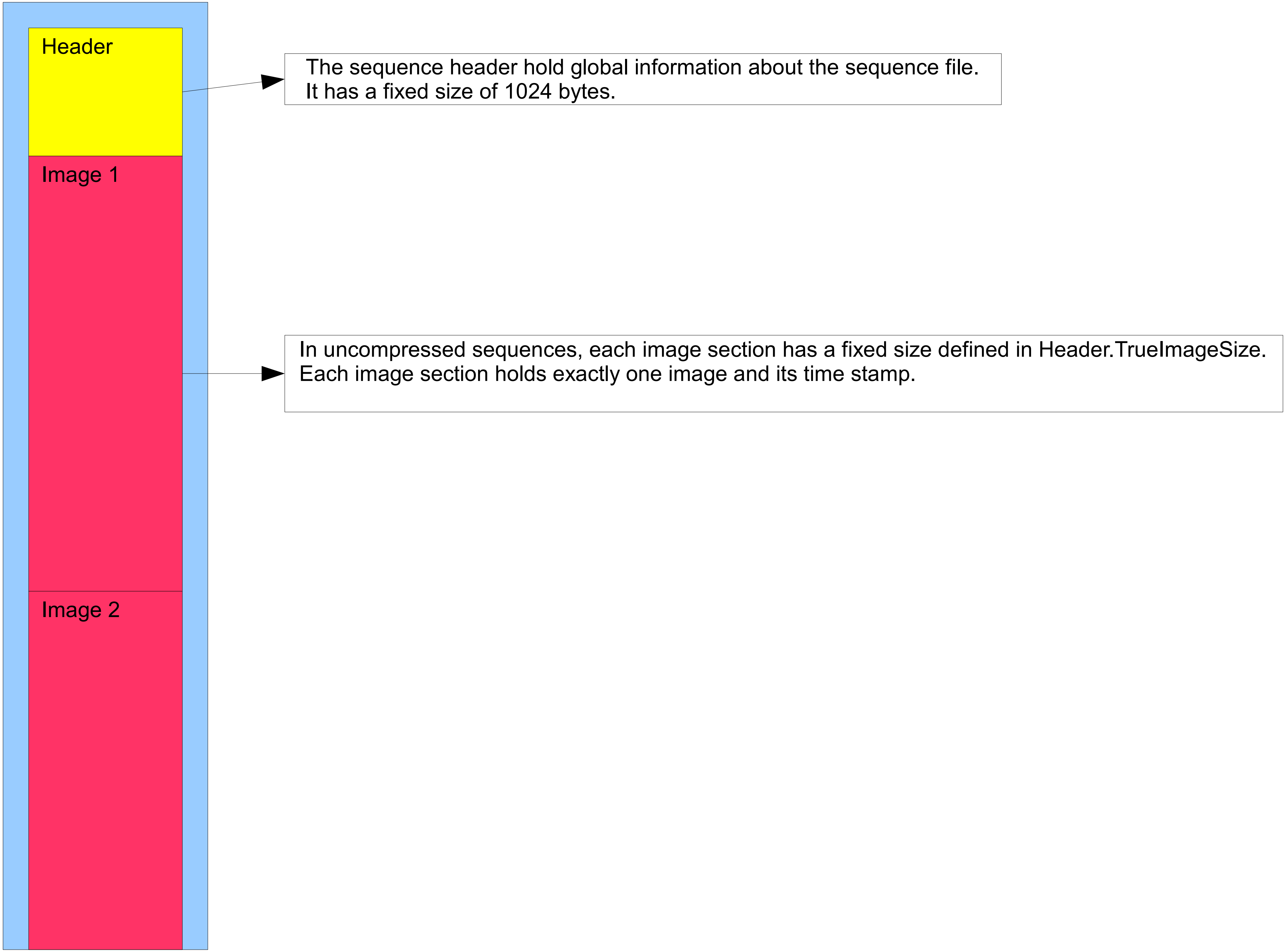
The image offset for image at index i will be : 8192 + (i * *TrueImageSize)*

The timestamp information is located at : 8192 + (i * *TrueImageSize)* + *ImageInfo.ImageSizeBytes.*

More

If you need more information or help retreiving data from our sequence files please contact us at support@norpix.com.

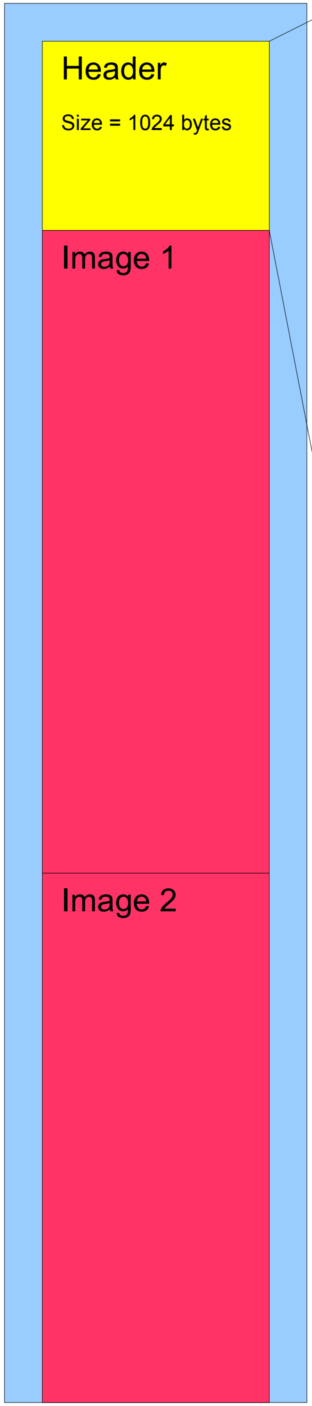
Sequence File Sections



The sequence header hold global information about the sequence file.
It has a fixed size of 1024 bytes.

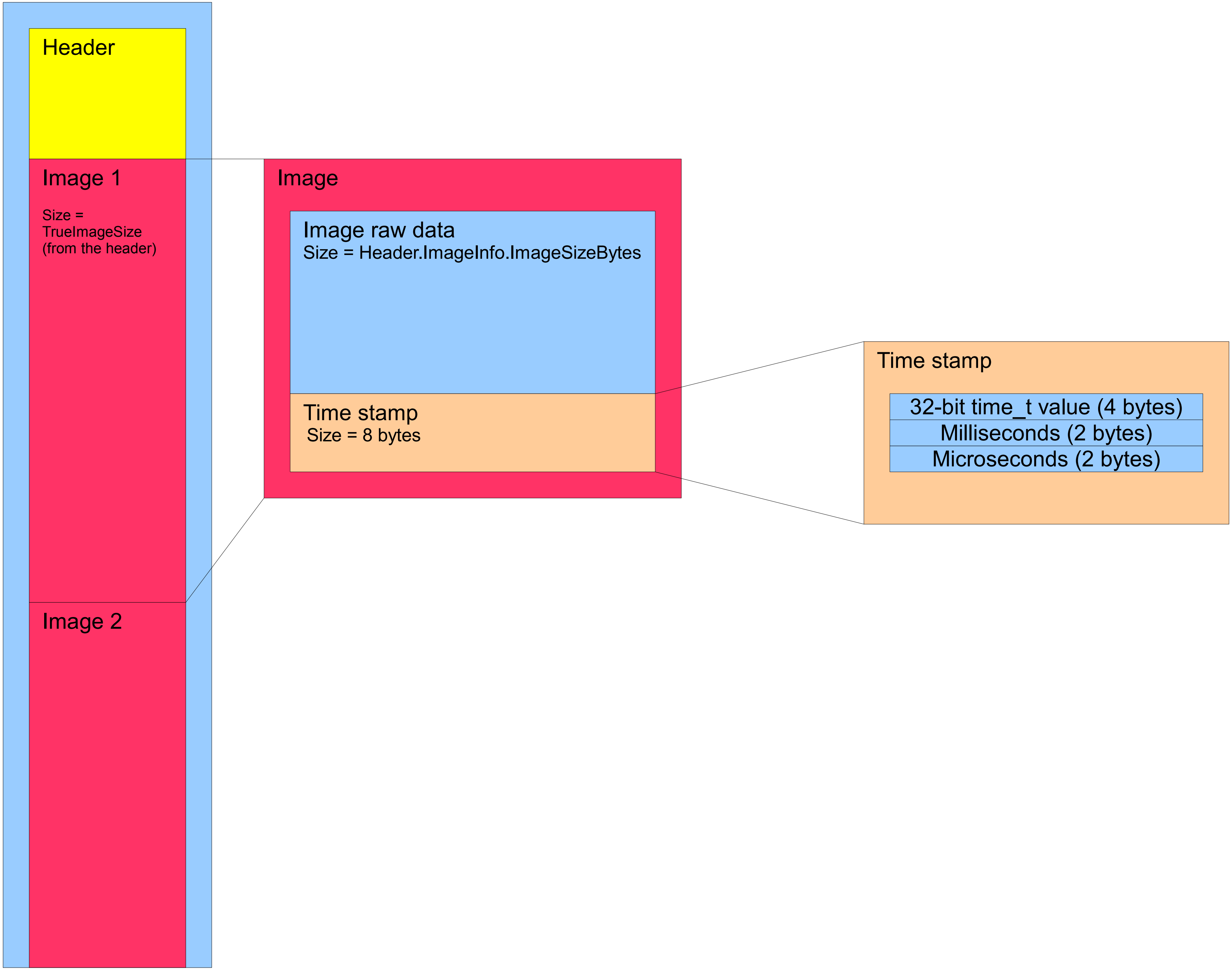
In uncompressed sequences, each image section has a fixed size defined in Header.TrueImageSize.
Each image section holds exactly one image and its time stamp.

Sequence File Sections



Name	Content	Offset [size]
long MagicNumber	Always 0xFEED	0 [4]
wchar_t Name[12]	Always "Norpix *seq*\n"	4 [24]
long Version	Sequence Header Version	28[4]
long HeaderSize	Always 1024	32 [4]
BYTE Description[512]	User description	36 [512]
CImageInfo ImageInfo	See belows for a description of the CImageInfo struct	548 [24]
unsigned long AllocatedFrames	Number of frames allocated in the sequence	572 [4]
unsigned long Origin	Should be 0 if not Pre/Post recorded	576 [4]
unsigned long TrueImageSize	Number of bytes between the first pixel of each successive images	580 [4]
double FrameRate	Suggested Frame rate for playback (in fps)	584 [8]
long DescriptionFormat	The content of "Description" 0-unicode 1-ascii 2-data	592 [4]
ULONG ReferenceFrame	ReferenceFrame index (0 if none)	596 [4]
ULONG FixedSize	Fixed size for compressed sequences (0 if none)	600 [4]
ULONG Flags	NorPix reserved flags	604 [4]
long BayerPattern	Bayer pattern used	608 [4]
long TimeOffsetUS	Time offset applied to each image timestamp	612 [4]
long ExtendedHeaderSize	Size of the extended header (if any)	616 [4]
eHCompression CompressionFormat	The compression used	620 [4]
long ReferenceTime	Custom Reference Time (time_t format)	624 [4]
ushort ReferenceTimeMS	Custom Reference Time (milliseconds part)	628 [2]
ushort ReferenceTimeUS	Custom Reference Time (microseconds part)	630 [2]
ulong H264GOP	Group of Picture value if H264 compression is used	632 [4]
ulong H264Bitrate	Bitrate if H264 compression is used	636 [4]
unsigned long JPEGQualityInfo	JPEG Format Quality and lossless	640 [4]
eHImageFormat H264DecodeFormat	H264 decode format	644 [4]
long long IndexOffset	Offset of compression index data	648 [8]
unsigned long OldestFrameIndex	Index of the oldest frame (will be > 0 if loop recording)	656 [4]
unsigned long BytesAlignment	Image alignement in bytes (for uncompressed sequences)	660 [4]
BYTE Padding[360]	Unused bytes, reserved for future uses.	664 [360]

Sequence File Sections



Header

Image 1

Size = TrueImageSize (from the header)

Image 2

Image

Image raw data
Size = Header.ImageInfo.ImageSizeBytes

Time stamp
Size = 8 bytes

Time stamp

32-bit time_t value (4 bytes)
Milliseconds (2 bytes)
Microseconds (2 bytes)

Additionnal Structures and Enums (1/2)

```
struct CImageInfo
{
   unsigned long ImageWidth;       //Image width in pixel
   unsigned long ImageHeight;      //Image height in pixel
   unsigned long ImageBitDepth;    //Image depth in bits (8,16,24,32)
   unsigned long ImageBitDepthReal;//Precise Image depth (x bits)
   unsigned long ImageSizeBytes;   //Size used to store one image.
   eHImageFormat ImageFormat;      //See formats below
};
```

```
enum eHCompression
{
        H_COMPRESSION_NONE=0,
        H_COMPRESSION_JPEG,
        H_COMPRESSION_RLE,
        H_COMPRESSION_HUFFMAN,
        H_COMPRESSION_LZ,
        H_COMPRESSION_RLE_FAST,
        H_COMPRESSION_HUFFMAN_FAST,
        H_COMPRESSION_LZ_FAST,
        H_COMPRESSION_H264,
        H_COMPRESSION_WAVELET
};
```

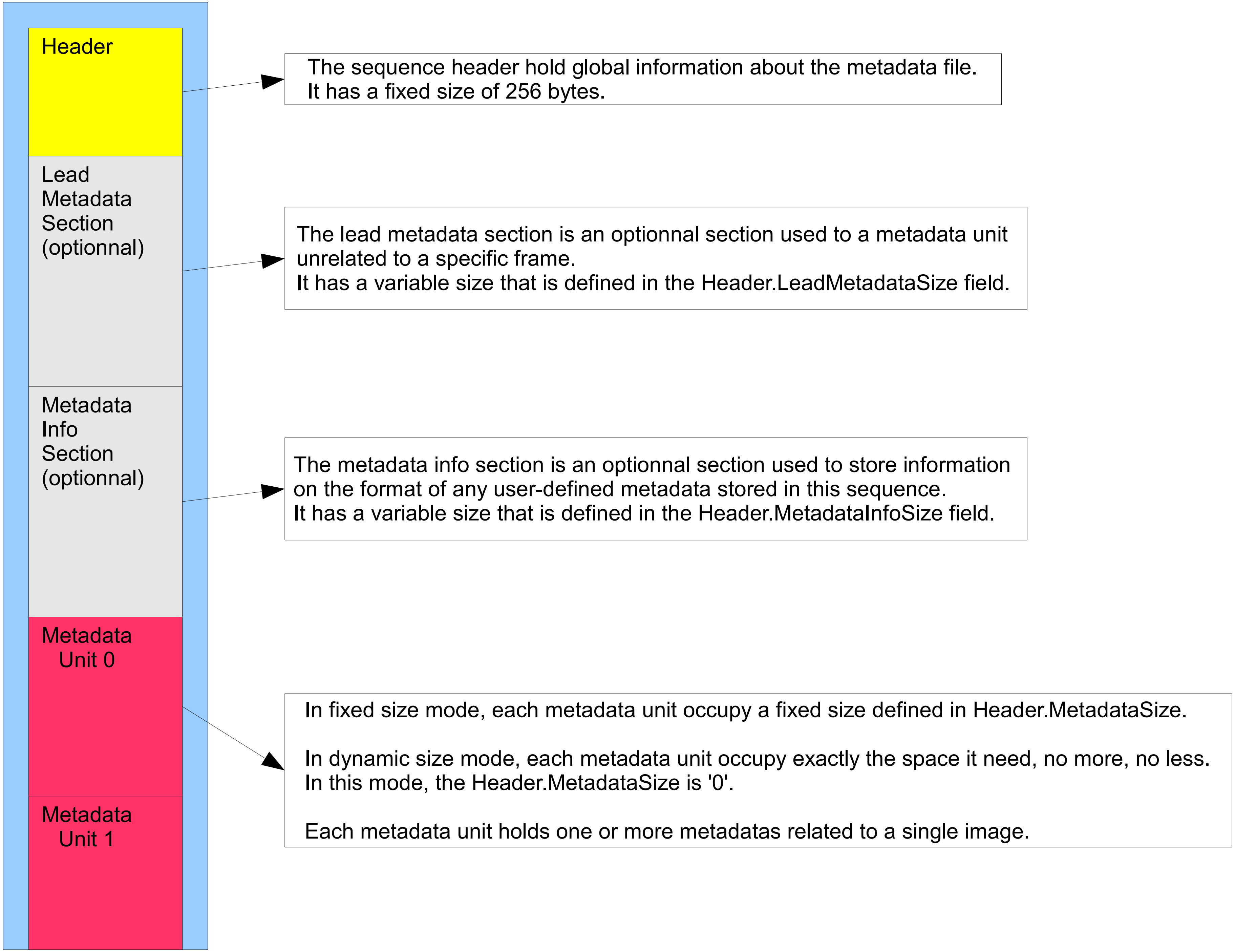
```
enum eHMetadataFormat
{
    H_METADATA_UNKNOWN = 0,
    H_METADATA_BOOL,    //bool (1 byte)
    H_METADATA_BYTE,    //byte (1 byte)
    H_METADATA_SHORT,   //short (2 bytes)
    H_METADATA_USHORT,  //unsigned short (2 bytes)
    H_METADATA_INT,     //int (4 bytes)
    H_METADATA_UINT,    //unsigned int (4 bytes)
    H_METADATA_DOUBLE,  //double (8 bytes)
    H_METADATA_STRING,  //wchar_t[] (variable)
    H_METADATA_BINARY   //BYTE[] (variable)
    H_METADATA_INT64,   //__int64 (8 bytes)
    H_METADATA_UINT64   //unsigned __int64 (8 bytes)
};
```

```
enum eHImageFormat
{
    H_IMAGE_UNKNOWN = 0,                  //Unknown format
    H_IMAGE_MONO = 100,                   //Monochrome Image (LSB)
    H_IMAGE_MONO_BAYER = 101,             //Raw Bayer Image (treated as H_IMAGE_MONO)
    H_IMAGE_BGR = 200,                    //BGR Color Image
    H_IMAGE_PLANAR = 300,                 //Planar Color Image
    H_IMAGE_RGB = 400,                    //RGB Color Image
    H_IMAGE_BGRx = 500,                   //BGRx Color Image
    H_IMAGE_YUV422 = 600,                 //YUV422
    H_IMAGE_YUV422_20 = 610,
    H_IMAGE_UVY422 = 700,                 //UVY422
    H_IMAGE_UVY411 = 800,                 //UVY411
    H_IMAGE_UVY444 = 900,                 //UVY444
    H_IMAGE_BGR555_PACKED = 905,          //PhynxRGB
    H_IMAGE_BGR565_PACKED = 906,
    H_IMAGE_MONO_MSB = 112,               //Only for > 8 bit per pixel, MSB align litle endian 10 bit: JIHGFEDC BA000000
    H_IMAGE_MONO_BAYER_MSB = 113,         //Only for > 8 bit per pixel, MSB align
    H_IMAGE_MONO_MSB_SWAP = 114,          //Only for > 8 bit per pixel, MSB align big endian 10 bit: BA000000 JIHGFEDC
    H_IMAGE_MONO_BAYER_MSB_SWAP = 115,    //Only for > 8 bit per pixel, MSB align
    H_IMAGE_BGR10_PPACKED = 123,          //Only for 10 bit per pixel, LSB align
    H_IMAGE_BGR10_PPACKED_PHOENIX = 124,  //Only for 10 bit per pixel, LSB align, RRRRRRRR RR00GGGG GGGGGGBB BBBBBBBB
    H_IMAGE_RGB10_PPACKED_PHOENIX = 125,  //Only for 10 bit per pixel, LSB align, BBBBBBBB BB00GGGG GGGGGGRR RRRRRRRR
    H_IMAGE_MONO_PPACKED = 131,           //Only for > 8 bit per pixel, MSB align
    H_IMAGE_MONO_BAYER_PPACKED = 132,     //Only for > 8 bit per pixel, MSB align
    H_IMAGE_MONO_PPACKED_8448 = 133,      //Only for > 8 bit per pixel, MSB align
    H_IMAGE_MONO_BAYER_PPACKED_8448 = 134,//Only for > 8 bit per pixel, MSB align
    H_IMAGE_GVSP_BGR10V1_PACKED = 135,    //Only for  10 bit per pixel(From Gige Vision) BBBBBBBB GGGGGGGG RRRRRRRR 00BBGGRR
    H_IMAGE_GVSP_BGR10V2_PACKED = 136,    //Only for  10 bit per pixel(From Gige Vision)00BBBBBB BBGGGGGG GGGGRRRR RRRRRRRR
    H_IMAGE_BASLER_VENDOR_SPECIFIC = 1000,
    H_IMAGE_EURESYS_JPEG = 1001,
    H_IMAGE_ISG_JPEG = 1002
};
```

Additionnal Structures and Enums (2/2)

```
enum eHCompression
{
    H_COMPRESSION_NONE=0,                           //Uncompressed sequence
    H_COMPRESSION_JPEG=1,                           //JPEG compressed images
    H_COMPRESSION_RLE=2,                            //RLE compressed images (stp3 algo)
    H_COMPRESSION_HUFFMAN=3,                        //HUFFMAN compressed images (stp3 algo)
    H_COMPRESSION_LZ=4,                             //LZ compressed images (stp3 algo)
    H_COMPRESSION_RLE_FAST=5,                       //RLE Fast compressed images (ippwrapper)
    H_COMPRESSION_HUFFMAN_FAST=6,                   //HUFFMAN  Fast compressed images (ippwrapper)
    H_COMPRESSION_LZ_FAST=7,                        //LZ Fast compressed images (ippwrapper)
    H_COMPRESSION_H264=8,                           //H264 compressed images
    H_COMPRESSION_WAVELET=9,                        //Wavelet compressed images
    H_COMPRESSION_H264_RGB = 16,                    //H264 compressed images, RGB after uncompressed (specific from some cameras)
    H_COMPRESSION_JPEG_LOSSESS = 17,                //JPeg lossless jpeg compressed SEQ
    H_COMPRESSION_CAYER = 18,                       //Createc cayer compression.
};
```

Metadata File Sections



Header

Lead Metadata Section (optionnal)

Metadata Info Section (optionnal)

Metadata Unit 0

Metadata Unit 1

The sequence header hold global information about the metadata file.
It has a fixed size of 256 bytes.

The lead metadata section is an optionnal section used to a metadata unit unrelated to a specific frame.
It has a variable size that is defined in the Header.LeadMetadataSize field.

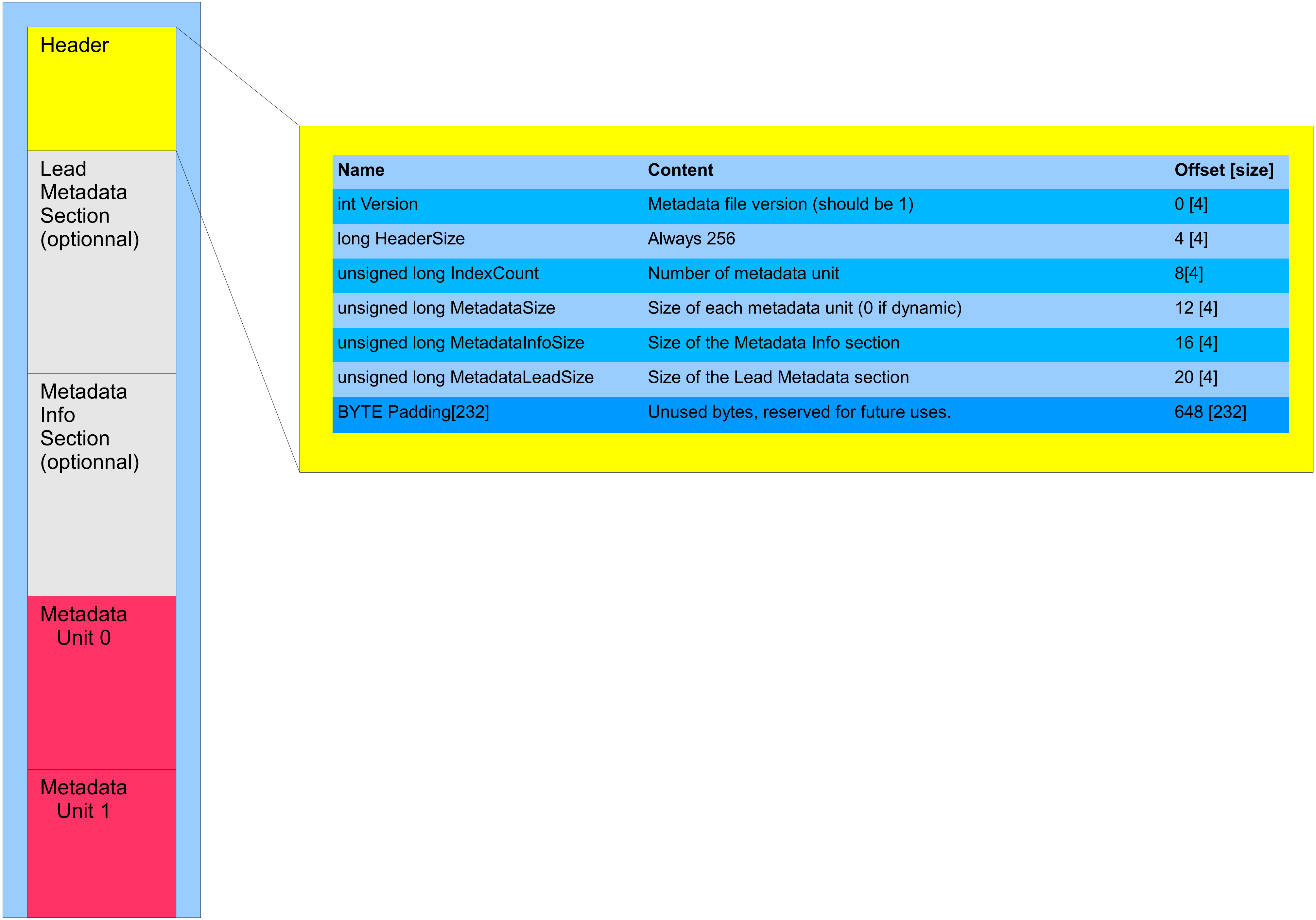
The metadata info section is an optionnal section used to store information on the format of any user-defined metadata stored in this sequence.
It has a variable size that is defined in the Header.MetadataInfoSize field.

In fixed size mode, each metadata unit occupy a fixed size defined in Header.MetadataSize.

In dynamic size mode, each metadata unit occupy exactly the space it need, no more, no less.
In this mode, the Header.MetadataSize is '0'.

Each metadata unit holds one or more metadatas related to a single image.

Metadata File Sections



Header

Lead Metadata Section (optionnal)

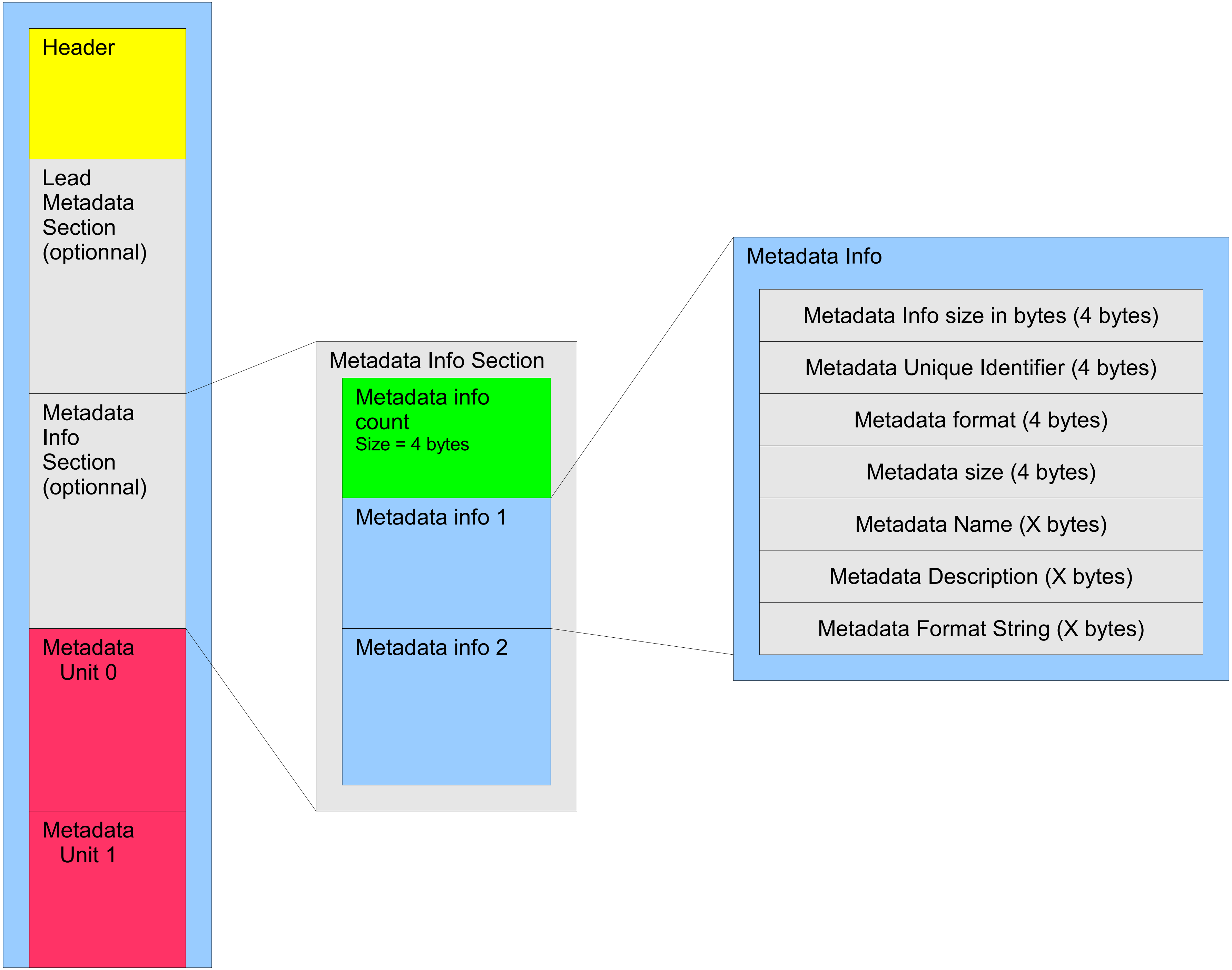
Metadata Info Section (optionnal)

Metadata Unit 0

Metadata Unit 1

Name	**Content**	**Offset [size]**
int Version	Metadata file version (should be 1)	0 [4]
long HeaderSize	Always 256	4 [4]
unsigned long IndexCount	Number of metadata unit	8[4]
unsigned long MetadataSize	Size of each metadata unit (0 if dynamic)	12 [4]
unsigned long MetadataInfoSize	Size of the Metadata Info section	16 [4]
unsigned long MetadataLeadSize	Size of the Lead Metadata section	20 [4]
BYTE Padding[232]	Unused bytes, reserved for future uses.	648 [232]

Metadata File Sections



Header

Lead Metadata Section (optionnal)

Metadata Info Section (optionnal)

Metadata Unit 0

Metadata Unit 1

Metadata Info Section

Metadata info count
Size = 4 bytes

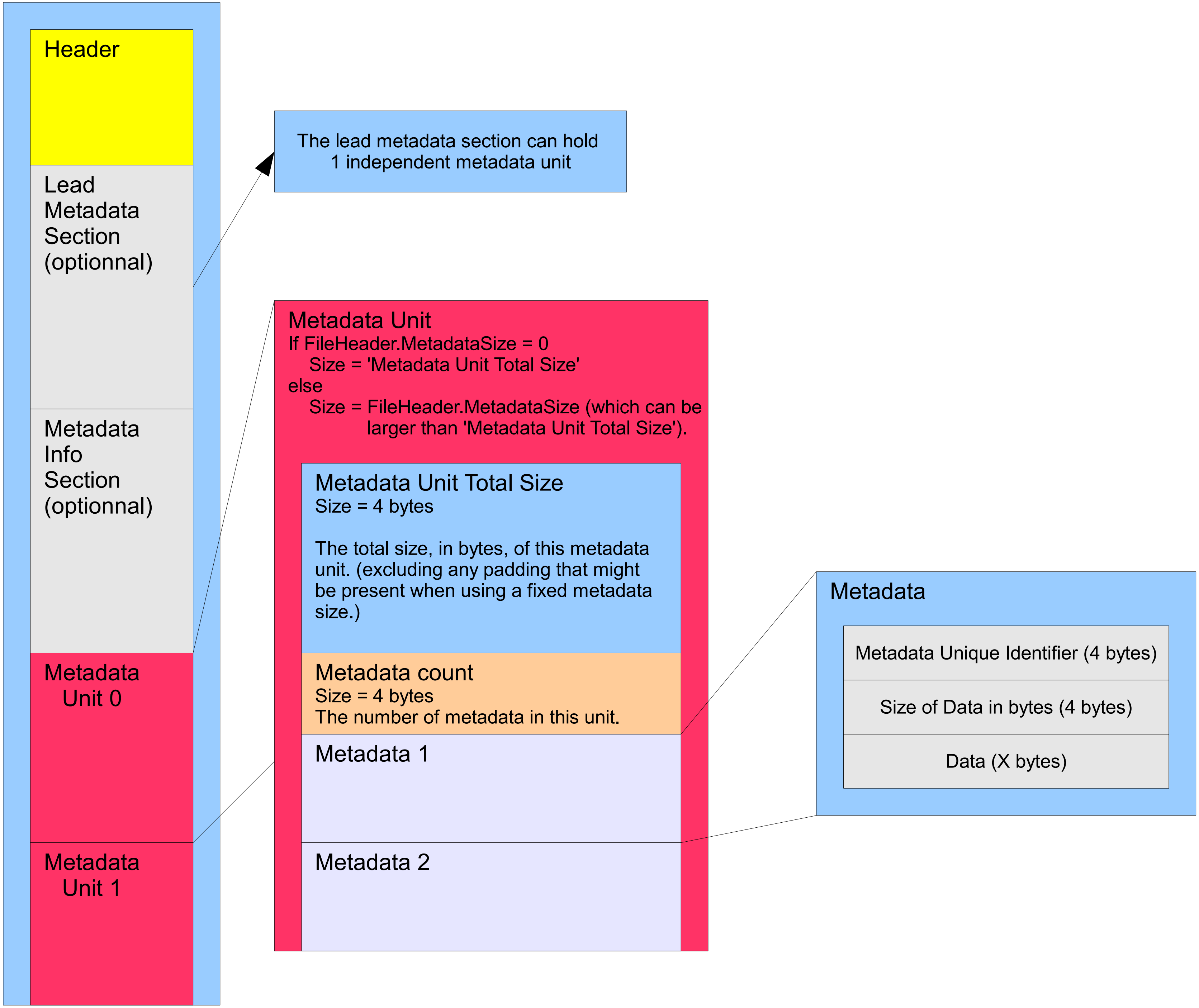
Metadata info 1

Metadata info 2

Metadata Info

- Metadata Info size in bytes (4 bytes)
- Metadata Unique Identifier (4 bytes)
- Metadata format (4 bytes)
- Metadata size (4 bytes)
- Metadata Name (X bytes)
- Metadata Description (X bytes)
- Metadata Format String (X bytes)

Metadata File Sections



Header

Lead Metadata Section (optionnal)

Metadata Info Section (optionnal)

Metadata Unit 0

Metadata Unit 1

The lead metadata section can hold 1 independent metadata unit

Metadata Unit

```
If FileHeader.MetadataSize = 0
    Size = 'Metadata Unit Total Size'
else
    Size = FileHeader.MetadataSize (which can be
           larger than 'Metadata Unit Total Size').
```

Metadata Unit Total Size

Size = 4 bytes

The total size, in bytes, of this metadata unit. (excluding any padding that might be present when using a fixed metadata size.)

Metadata count

Size = 4 bytes

The number of metadata in this unit.

Metadata 1

Metadata 2

Metadata

Metadata Unique Identifier (4 bytes)

Size of Data in bytes (4 bytes)

Data (X bytes)

How to set the Metadata size

Fixed size mode

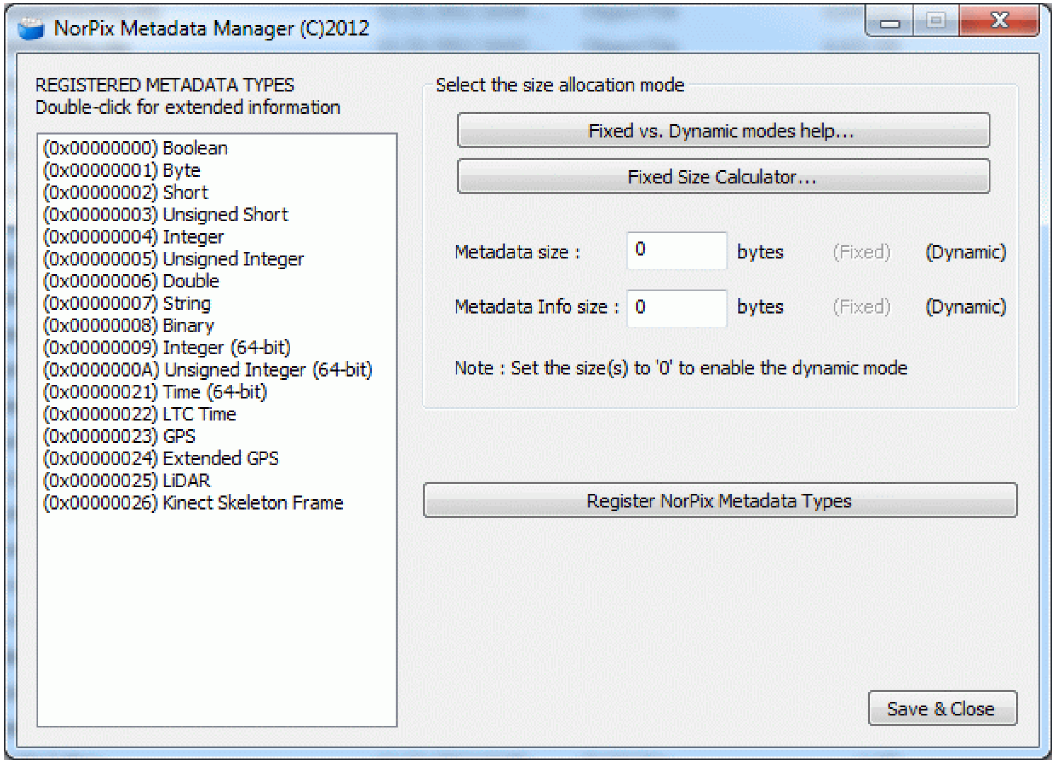
In this mode, the metadata file allocate a fixed size for each metadata unit and a fixed size for the metadata info section. Having a fixed size means that an image's metadata might not be saved to the file if the fixed size is not large enough. It can also lead to waste of space if the fixed size is much larger than what is needed to store the metadata after each image. This mode should only be used when recording in loops.

Dynamic size mode

In the dynamic mode, the metadata file allocate exactly the requiered space needed to store each metadata unit. The metadata info space is allocated by looking at the metadata bundled in the first frame to be written to the sequence. Dynamic mode ensure that you will have exactly the required size to store the metadata (and the metadata info, if any).

Metadata Manager

The Metadata Manager is a stand-alone application that can be used to configure which mode to use and the sizes of the sections. The Metadata Manager also offer a Size Calculator tool that allows you to estimate the required size for the "Metadata" and the "Metadata Info" sections. Simply select the metadata types that are to be saved with each image and the will do the rest.


NorPix Metadata Manager (C)2012
Fixed vs. Dynamic modes help...
Fixed Size Calculator...
Register NorPix Metadata Types
Save & Close

REGISTERED METADATA TYPES
Double-click for extended information

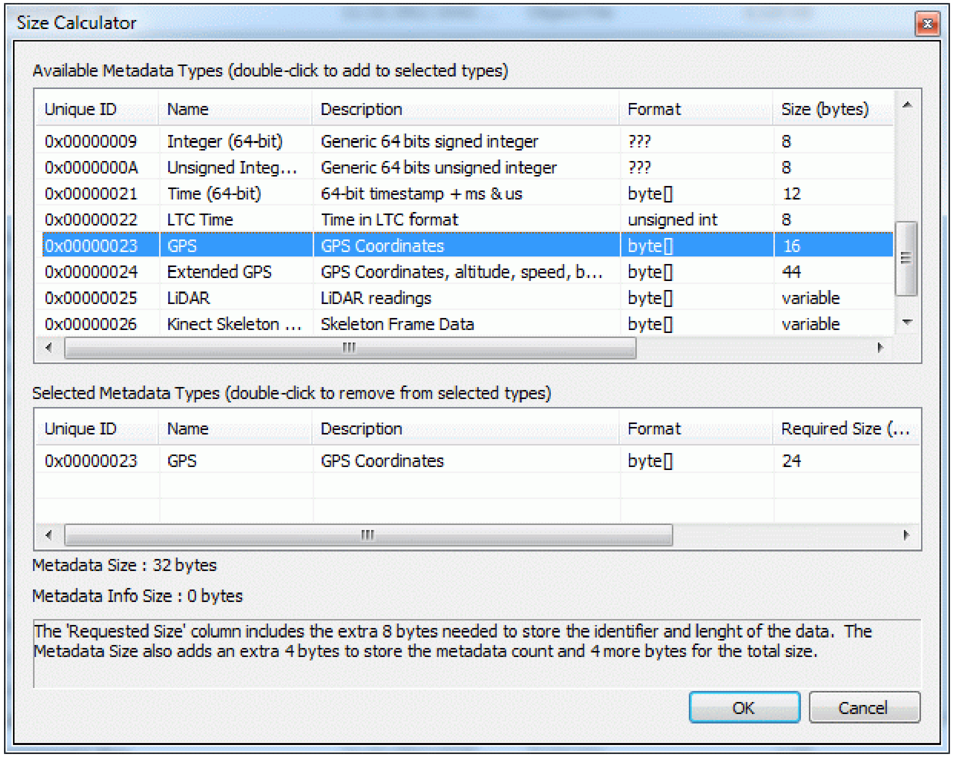
(0x00000000) Boolean
(0x00000001) Byte
(0x00000002) Short
(0x00000003) Unsigned Short
(0x00000004) Integer
(0x00000005) Unsigned Integer
(0x00000006) Double
(0x00000007) String
(0x00000008) Binary
(0x00000009) Integer (64-bit)
(0x0000000A) Unsigned Integer (64-bit)
(0x00000021) Time (64-bit)
(0x00000022) LTC Time
(0x00000023) GPS
(0x00000024) Extended GPS
(0x00000025) LiDAR
(0x00000026) Kinect Skeleton Frame

Select the size allocation mode

Metadata size : 0 bytes (Fixed) (Dynamic)

Metadata Info size : 0 bytes (Fixed) (Dynamic)

Note : Set the size(s) to '0' to enable the dynamic mode



Available Metadata Types (double-click to add to selected types)

Unique ID	Name	Description	Format	Size (bytes)
0x00000009	Integer (64-bit)	Generic 64 bits signed integer	???	8
0x0000000A	Unsigned Integ...	Generic 64 bits unsigned integer	???	8
0x00000021	Time (64-bit)	64-bit timestamp + ms & us	byte[]	12
0x00000022	LTC Time	Time in LTC format	unsigned int	8
0x00000023	GPS	GPS Coordinates	byte[]	16
0x00000024	Extended GPS	GPS Coordinates, altitude, speed, b...	byte[]	44
0x00000025	LiDAR	LiDAR readings	byte[]	variable
0x00000026	Kinect Skeleton ...	Skeleton Frame Data	byte[]	variable

Selected Metadata Types (double-click to remove from selected types)

Unique ID	Name	Description	Format	Required Size (...
0x00000023	GPS	GPS Coordinates	byte[]	24

Metadata Size : 32 bytes

Metadata Info Size : 0 bytes

The 'Requested Size' column includes the extra 8 bytes needed to store the identifier and lenght of the data. The Metadata Size also adds an extra 4 bytes to store the metadata count and 4 more bytes for the total size.