TrueBeam .xim image format

For questions, please send us an email at: TrueBeamDeveloper@varian.com

XIM images are binary files. Generally, XIM images have 4 main sections in the following order:

1. **Header:** 32 bytes, fixed size

File Format Identifier	Char, 8 bytes
File Format Version	Int4
Image Width in pixels	Int4
Image Height in pixels	Int4
Bits Per Pixel	Int4
Bytes Per Pixel	Int4
Compression indicator	Int4 (0=Uncompressed, 1=HND compression)

2. **Pixel Data:**

A. Uncompressed Data (Only if *Compression Indicator* =0):

Uncompressed Pixel Buffer Size	Int4 (= Image Width * Image Height * Bytes Per Pixel)
Uncompressed Pixel Buffer	Char1 x Uncompressed Pixel Buffer Size. These are the uncompressed pixel values

B. HND Compressed Data (Only if *Compression Indicator =1*):

Lookup Table Size	Int4
Lookup Table	Char1 x Lookup Table Size
Compressed Pixel Buffer Size	Int4
Compressed Pixel Buffer	Char1 x Compressed Pixel Buffer Size
Uncompressed Pixel Buffer Size	Int4 (= Image Width * Image Height * Bytes Per Pixel)

3. **Histogram:** optional

Number of Bins in Histogram	Int4 (0 if no histogram) -- 1024 is typical for XI
Histogram	Int4 x Number of Bins in Histogram

4. **Properties:** optional

Number of Properties	Int4 (=0 if no properties)

Properties:	
Property Name length	Int4 String length
Property Name	Char1 X Property Name Length
Property Type	Int4, 0 = Integer, 1 = Double, 2 = String, 4 = Double Array, 5 = Integer Array
Property Value	Per Property Type

Notes:

a) Multi-byte data types are stored as Little-endian, i.e. least-significant byte first. Images are gray scale from 8 to 32 bits/pixel.
b) The properties are not necessarily ordered.
c) Image orientation is in beam's eye view with pixel order top left to bottom right. The first row defines a line parallel to the plane of gantry motion.
d) Pixel data are encoded using slope and intercept. To convert, use the relationship: Pixel Value = (Stored Pixel Value * Slope) + Intercept

HND Compression/Decompression:

HND compression is a lossless compression algorithm. The compression algorithm exploits the typically prevalent similarity of neighboring pixel values by storing differences only instead of the actual pixel values.

HND compression applies only to 2 and 4 bytes/pixel images. Images with 1 byte/pixel are always stored uncompressed.

The XIM header contains Image Width, Image Height and Bytes Per Pixel, which are required to interpret the compressed pixel data.

The first row and the first pixel of the second row are stored uncompressed. The remainders of the pixels are compressed by storing only the difference between neighboring pixels.

E.g. consider the following hypothetical 12 pixel image:

R_{11}	R_{12}	R_{13}	R_{14}
R_{21}	R_{22}	R_{23}	R_{24}
R_{31}	R_{32}	R_{33}	R_{34}

Pixels R_{11} through R_{14} and R_{21} are stored uncompressed, while pixels R_{22} through R_{34} are compressed by storing only the difference: $diff = R_{11} + R_{22} - R_{21} - R_{12}$

Exploiting the fact that most images exhibit similarity in neighboring pixel values, the above difference can be stored using fewer bytes, e.g. 1, 2 or 4 bytes. Thus, the smoother the pixel values in the image the higher the compression ratio.

For decompression, the algorithm needs to know the byte size of each stored difference. To accomplish this, a lookup table is placed at the beginning of the image. The lookup table contains a 2-bit flag for each pixel which defines the byte size for each compressed pixel difference. So a flag value of 0 means the difference fits into one byte while 1 and 2 mean a two and four byte difference respectively.

Note: Since this lookup table represents additional storage overhead, a compressed image containing high frequency noise (and thus prevalent differences between neighboring pixel values) may end up being larger than the uncompressed image. In that case the image is typically stored uncompressed.

Thus, the size of the lookup table depends only on image size and can be computed using the expression: Lookup table size = [<Image Width> * (<Image Height> – 1)] / 4 + 0.5

The -1 accounts for the fact that the first row is stored uncompressed and thus requires no entries in the lookup table. The 4 denominator represents the fact that each lookup table entry needs two bits, thus four lookup table entries can be stored per byte. Finally 0.5 is added to round up to the next integer value.

Note: What about the first pixel of the second row that is also uncompressed?
Lookup table doesn't contain any flag corresponding to this pixel.

Note: Dimensionality of the image can result in unused 2-bit flag fields in the last byte of the lookup table. The number of unused flag fields can be determined by mod([<Image Width> * (<Image Height-1> – 1)], 4) where mod denotes the mathematical modulus operator. Therefore, last byte in the lookup table can contain 1-3 unused flags (i.e., 2 to 6 bits)