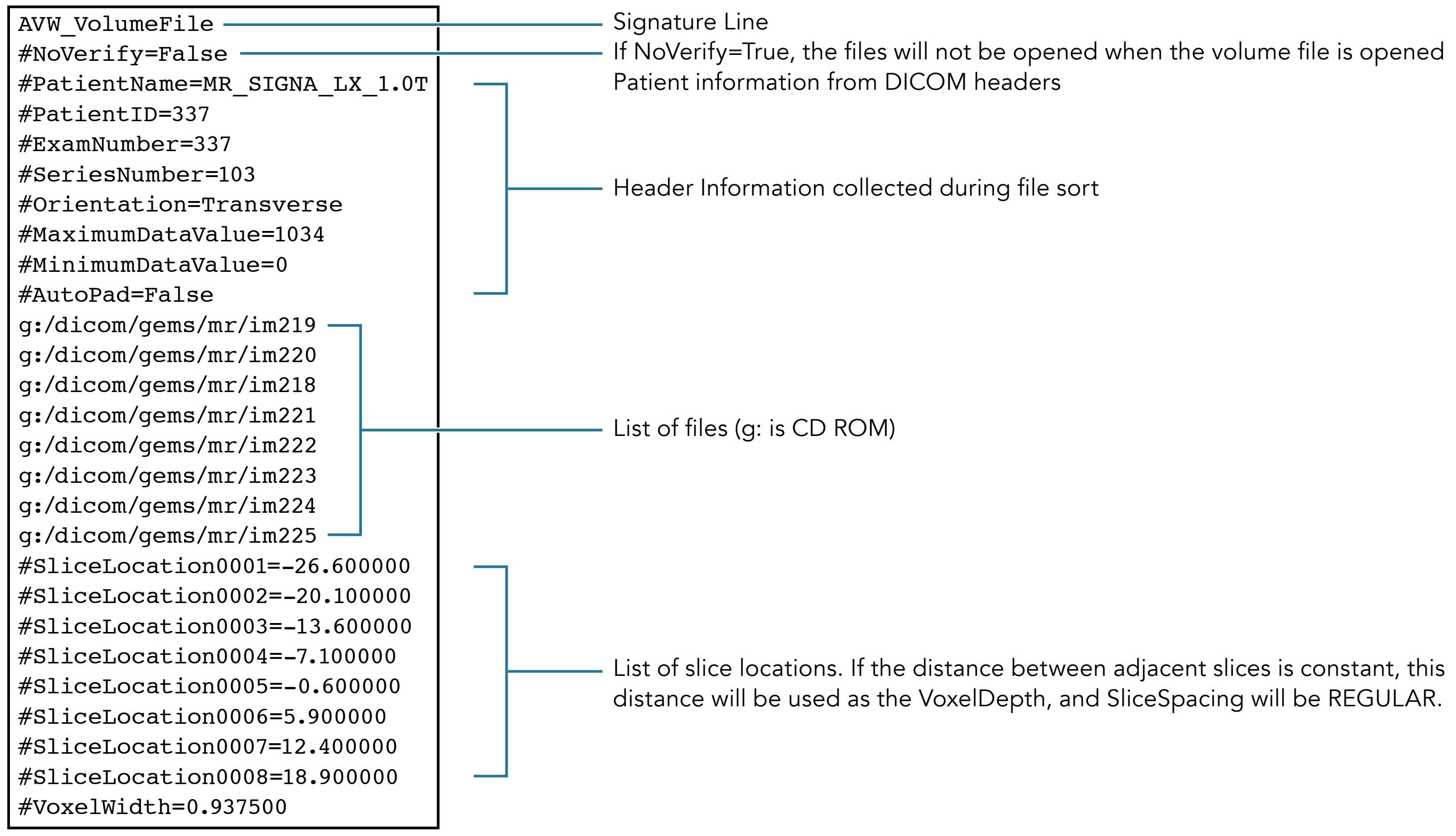
Table of Contents

Organizing a List of DICOM Files

AVW_Volume files are text files used to organize lists of 2D files for use with 3D applications in Analyze. When AVW_VolumeFiles are opened, native header information from the referenced files is returned into the Info string. AVW_VolumeFiles may also be annotated with additional header information. AVW_VolumeFiles can be created in Analyze by several tools in the ImportExport program, including Volume Tool, File Sort Tool, Raw Data Tool, Tape Tool and DICOMDIR tool. Analyze modules respond to the request to save a volume or series of images to a 2D limited format by creating a 'wrapper' volume file and writing each image to sequentially numbered files. The structure of an AVW_VolumeFile is simple. They can be easily created with a text editor. Lines beginning with # are taken as information tags. Standard tags used by Analyze are:

Example AVW_VolumeFile



```
AVW_VolumeFile
#NoVerify=False
#PatientName=MR_SIGNA_LX_1.0T
#PatientID=337
#ExamNumber=337
#SeriesNumber=103
#Orientation=Transverse
#MaximumDataValue=1034
#MinimumDataValue=0
#AutoPad=False
g:/dicom/gems/mr/im219
g:/dicom/gems/mr/im220
g:/dicom/gems/mr/im218
g:/dicom/gems/mr/im221
g:/dicom/gems/mr/im222
g:/dicom/gems/mr/im223
g:/dicom/gems/mr/im224
g:/dicom/gems/mr/im225
#SliceLocation0001=-26.600000
#SliceLocation0002=-20.100000
#SliceLocation0003=-13.600000
#SliceLocation0004=-7.100000
#SliceLocation0005=-0.600000
#SliceLocation0006=5.900000
#SliceLocation0007=12.400000
#SliceLocation0008=18.900000
#VoxelWidth=0.937500
```

Signature Line

If NoVerify=True, the files will not be opened when the volume file is opened

Patient information from DICOM headers

Header Information collected during file sort

List of files (g: is CD ROM)

List of slice locations. If the distance between adjacent slices is constant, this distance will be used as the VoxelDepth, and SliceSpacing will be REGULAR.

#NoVerify — if False all files in the list are opened for verification
#AutoPad — It True allows files to be different sizes, each image is padded into an image as wide as the widest image in the list and as high as the tallest image in the list.
#VoxelWidth — voxel dimension in X
#VoxelHeight — voxel dimension in Y
#VoxelDepth — voxel depth in Z, or space between slices
#VoxelUnits — unit of measure for voxel dimensions
#MaximumDataValue — largest voxel value in data set
#MinimumDataValue — smallest voxel value in data set
#Orientation — anatomic orientation of slices; Transverse, Coronal, or Sagittal
#SliceLocation#### — slice location for slice #### .i.e SliceLocation0001
#SliceSpacing — REGULAR or IRREGULAR

You may annotate an AVW_VolumeFile with any other information you wish by include lines of the form:

#MyTag=MyValue — These tags or header elements will be ignored by Analyze but may be referenced in the application that you write using AVW.

The AVW_VolumeFile can also be used as a header which points to another file. Specifications in the AVW_VolumeFile control reading of data from the binary file into AVW_Images and AVW_

AVW_VolumeFile as a Header to a Single Binary File

Creating a volume file header for a binary file of an unsupported image file format provides a way of reading the image data without converting it into another copy of the data. The required information between `#RawDataDescriptionStart` and `#RawDataDescriptionEnd` can be generated in the Volume Tool of ImportExport by selecting the right mouse button option Describe Raw Data.

Example AVW_VolumeFile

```
AVW_VolumeFile
#NoVerify=False
#AutoPad=False
#RawDataDescriptionStart
#SecondaryDataFormat=RawData
#VoxelOffset=6144
#Width=256
#Height=256
#Depth=1
#DataType=AVW_SIGNED_SHORT
#ByteSwap=Pairs
#ReverseBits=No
#FlipX=No
#FlipY=Yes
#RawDataDescriptionEnd

H:/Medtronic/1130-40-45.ima
```

Signature Line
If NoVerify=True, the files will not be opened when the volume file is opened
Patient information from DICOM headers

All tag lines between `#RawDataDescriptionStart` and `#RawDataDescriptionEnd` are the same as what appears in the Raw Data Tool in ImportExportList of files (g: is CD ROM)

Pointer to raw data file

Organizing a List of ‘Unknown’ Files

A list of unknown files can be treated as a 3D volume file if there is one image in each file and all files can be processed with the same parameters.

Example AVW_VolumeFile

```
AVW_VolumeFile
#NoVerify=False
#AutoPad=False
#RawDataDescriptionStart
#SecondaryDataFormat=RawData
#VoxelOffset=6144
#Width=256
#Height=256
#DataType=AVW_SIGNED_SHORT
#ByteSwap=Pairs
#ReverseBits=No
#FlipX=No
#FlipY=Yes
#VoxelWidth=.5000
#VoxelHeight=.5000
#VoxelDepth=1.5000
#RawDataDescriptionEnd

H:/Medtronic/1130-40-45.ima
H:/Medtronic/1130-40-46.ima
H:/Medtronic/1130-40-47.ima
H:/Medtronic/1130-40-48.ima
```

Signature Line

Note that Depth information is supplied by the file list and not specified in the RawDataDescriptionPointer to raw data file

List of raw data files

AnalyzeAVW Image File Format

The AnalyzeAVW image file format consists of a single file that begins with a self-documenting textual description of the data and the byte offset in the file at which the pixel data starts. No particular suffix or extension is used to identify these files with the exception of .mmap for memory-mapped files in the Analyze workspace.

Following is an example of textual information found in a memory mapped file:

```
AVW_ImageFile 1.00 4096
DataType=AVW_FLOAT
Width=128
Height=128
Depth=140
NumVols=1
ColormapSize=0
BeginInformation
DataFormat="AnalyzeAVW"
ExamDescription="WB SCAN"
ExamID="1.2.840.113619.2.64.672.944162198.11223344"
HospitalName="Mayo Clinic"
MaximumDataValue=0.121805
MinimumDataValue=-0.014635
OriginalDir="/images/demo/PET"
OriginalFile="GE_Body_MultiVolume"
OriginalFormat="GEADVANCE"
OriginalOrient="AVW_TRANSVERSE"
OriginalVolumes="1-4"
PatientID="Charlton5 ### ###"
PatientName="LastName, FirstName"
VoxelDepth=4.250000
VoxelHeight=4.296875
VoxelWidth=4.296875
EndInformation
MoreInformation=-1
Vol Slc Offset Length Cmp Format
.CONTIG
EndSliceTable
```

The first line contains the signature (AVW_ImageFile), the version number (1.00), and the byte offset to the image data in file (4096). To facilitate memory mapping voxel data across all platforms the byte offset value is always a multiple of 4 kilobytes (4096). The fields `Width, Height, Depth, NumVols`, and `ColormapSize` are all required by AVW and correspond to the same named elements of AVW_Images and AVW_Volumes. `Width` and `Height` are in voxels, `Depth` in slices, `NumVols` is the number of volumes in the file, and `ColormapSize` is the number of palette colors (if any) with this data. All of these tags correspond to structure elements of the same name in the *AVW_ImageFile*, *AVW_Image*, and *AVW_Volume* structures. If there were a colormap present it would immediately follow as a series if RGB color triples 1 to a line for each color in the palette.

Lines between BeginInformation and EndInformation are optional header information that the AVW ImageIO functions read from the proprietary headers when creating the file. AVW functions will use `MaximumDataValue` and `MinimumDataValue` in the scaling of data for display. Likewise `VoxelWidth`, `VoxelHeight`, and `VoxelDepth` are used for resizing the data as in interpolation of data to produce cubic voxels.

```
Vol Slc Offset Length Cmp Format
.CONTIG
EndSliceTable
```

The slice table indicates the byte offset of each slice in the file. Memory mapped files are uncompressed and contiguous, so the detail is unnecessary. Space between “EndSliceTable” and byte 4096 is filler and ignored. Textual descriptions grow in 4086 bytes increments if necessecary while a file is being written.

Images are stored as a contiguous byte stream. Images of odd width are not padded. Voxel values are stored BigEndian unless otherwise noted on the header.

The below example shows a potion of file that was written on a LittleEndian machine (Endian=Little) with ZLIB compression. Note the slice table shows the location and length of data for each compressed slice in the file. Note also that this file has 8192 bytes partitioned for textural description of the data. If this file is read on a “Big Endian” machine images must be byte swapped.

```
AVW_ImageFile 1.00 8192
   DataType=AVW_FLOAT
   Width=128
   Height=128
   Depth=140
   NumVols=1
   Endian=Little
   ColormapSize=0
BeginInformation
   DataFormat=”AnalyzeAVW”
   ExamDescription=”WB SCAN”
   ExamID=”1.2.840.113619.2.64.672.944162198.11223344”
   HospitalName=”Mayo Clinic”
   MaximumDataValue=0.121805
   MinimumDataValue=-0.014635
   OriginalDir=”/images/demo/PET”
   OriginalFile=”GE_Body_MultiVolume”
   OriginalFormat=”GEADVANCE”
   OriginalOrient=”AVW_TRANSVERSE”
   OriginalVolumes=”1-4”
   PatientID=”Charlton5 ### ###”
   PatientName=”LastName, FirstName”
   SliceNumber=139
   VoxelDepth=4.250000
   VoxelHeight=4.296875
   VoxelWidth=4.296875
EndInformation
MoreInformation=-1
Vol Slc Offset Length Cmp Format
   0  0  8192  41414  2
   0  1  49606  41060  2
   0  2  90666  40907  2
   0  3  131573  40655  2
   0  4  172228  39682  2
   0  5  211910  40838  2
   0  6  252748  39964  2
```

Colormapped Example

Colormapped image files are generally of data type AVW_UNSIGNED_CHAR. In this example voxels with a value of 0 are displayed with the first palette entry (32 32 128) *midnight blue*.

```
AVW_ImageFile                1.00     4096
   DataType=AVW_UNSIGNED_CHAR
   Width=340
   Height=340
   Depth=1
   NumVols=1
   Endian=Little
   ColormapSize=251
       32  32  128
       0  0  0
       4  4  4
       8  8  8
       12  12  12
       16  16  16
       20  20  20
       24  24  24
           …(continues for 251 entries)
BeginInformation
   BestThumbnail=0
   DataFormat="AnalyzeAVW"
EndInformation
MoreInformation=-1
Vol Slc Offset Length Cmp Format
   CONTIG
```

Image and Volume Order

In the voxel portion of the file images are stored as contiguous runs of voxels with no padding to accommodate odd image widths or odd byte counts. Volumes are stored as continuous runs of slices. Twenty-four bit color (RGB) images however are stored with a continuous RedVolume, followed by the Green and Blue Volumes. A file of 3 slices of 2 pixel square images is stored as follows:

`Byte >`

RRRR*RRRR*RRRR**GGGG***GGGG*GGGG**BBBB***BBBB*BBBB

RGB slice 1

RGB slice 2

RGB slice 3

Or taking each 4 bytes as a "slice" the slices are are ordered $R_1R_2R_3$ $G_1G_2G_3$ $B_1B_2B_3$

Multivolume RGB files follow the same pattern.

`Byte >`

Volume 1 Volume 2
RRRR*RRRR*RRRR**GGGG***GGGG*GGGG**BBBB***BBBB*BBBB**rrrr***rrrr*rrrr**gggg***gggg*gggg**bbbb***bbbb*bbbb

This AVW_ImageFIle format supports 2D, 3D, & 4D. Voxel data types of 8, 16, 32 signed and unsigned integers, floating point, complex, and 24 bit reg-green blue are supported. It is extensible to accommodate new data types, compression schemes and user defined header information. Developers should use AVW and the current version of the shared library libavwAnalyzeAVW to read and write these files as these specifications are open to change. The format is somewhat self-documenting and text and voxel sections are easily separated with Unix file utilities.