Imaris5.5 File Format Description (IMS)

The IMS format is used by the Imaris and ImarisViewer software from Oxford Instruments. ImarisViewer is a free 3D/4D microscopy image viewer for viewing raw images as well as those analyzed with Imaris. You can download ImarisViewer at https://imaris.oxinst.com/imaris-viewer. ImarisViewer comes with some demo images and it also has the capability to convert images from most microscopy image formats to the IMS format. This document describes the IMS format to enable readers to implement a file reader for images stored in IMS format or to implement a file writer to store images in IMS format.

1 The IMS Format For High Performance Visualization And Analysis

The IMS file format is designed to allow fast visualization and processing of very large 3D/4D images. For this purpose it stores not only the original image data but also lower resolution versions of the original, and it stores them in "small" contiguous 3D blocks[1]. This blockwise multi-resolution storage allows the visualization software to efficiently load only those 3D blocks required for each view. As shown in Figure 1 this can mean that the renderer loads different resolutions in different parts of the view. It can also mean that the renderer skips data outside of the current field of view. Or it can mean that the renderer loads only low resolution data when the zoom level is small. The blockwise multi-resolution storage provides the basis both for fast rendering of large images and for rendering of extremely large images that exceed a computer's RAM.

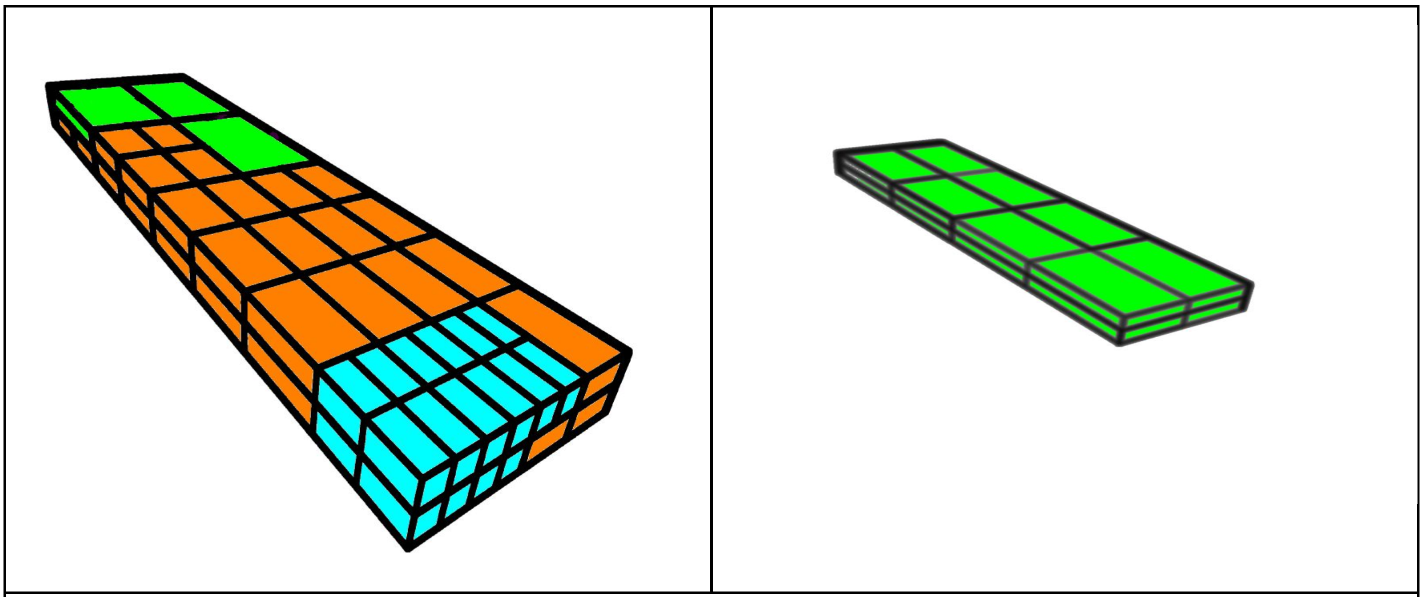


Figure 1. View dependent rendering depicting the 3D blocks loaded for the respective view. The colors depict resolution levels. The view on the left renders high resolution in the front of the scene and low resolution in the back. The view on the right renders low resolution data only.

2 IMS is based on HDF5

The IMS format is based on the HDF5 format from the HDFGroup (https://www.hdfgroup.org/). The HDF5 library for writing and reading of HDF5 files has proven to be very useful and well suited for storage of large

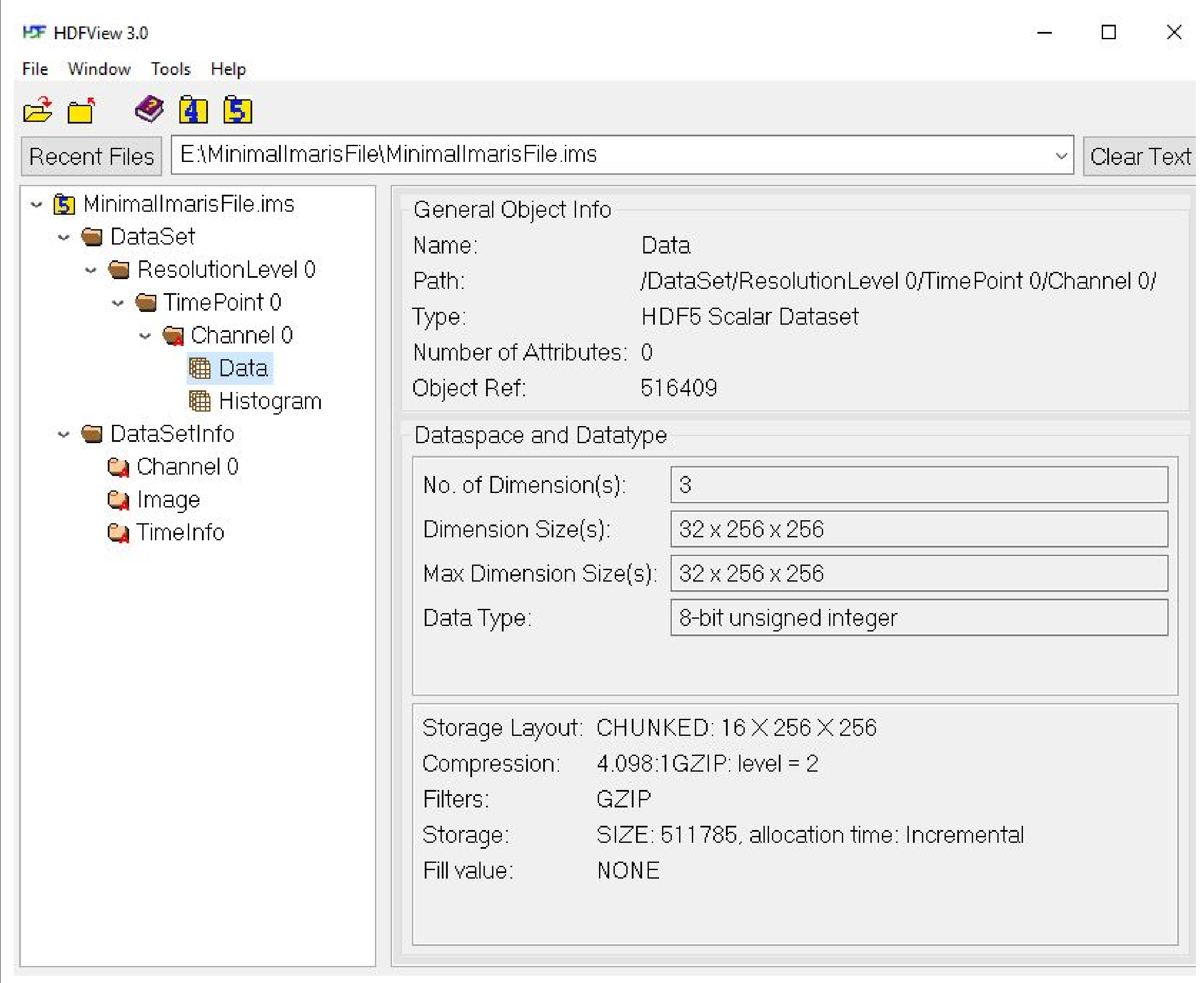
[1] Imaris typically uses blocks of 1MB size.

amounts of data. The HDF5 format and library already come with the capability to store big multidimensional datasets in blocks that are called “chunks” in HDF5 terminology. The IMS format piggybacks on this feature to achieve blockwise storage. The multi-resolution storage in the IMS format on the other hand is not natively HDF5 but is an IMS specific way of storing data in HDF5 that will be described in this document.

Throughout this document we will discuss the IMS format as a format on top of HDF5. We will discuss the format using HDF5 terminology and the HDF5 library. The implementation details of HDF5 itself are considered irrelevant for our purposes[2]

3 Viewing an IMS file with HDFView

To learn about the IMS format we recommend viewing an IMS file using the HDFView tool provided by the HDFGroup. In this tool the hierarchical structure of an IMS file typically looks similar to what is shown in Figure 2. With this tool it becomes relatively easy to understand what is contained within an IMS file.



```
MinimalImarisFile.ims
  DataSet
    ResolutionLevel 0
      TimePoint 0
        Channel 0
          Data
          Histogram
  DataSetInfo
    Channel 0
    Image
    TimeInfo
```

General Object Info

Name:	Data
Path:	/DataSet/ResolutionLevel 0/TimePoint 0/Channel 0/
Type:	HDF5 Scalar Dataset
Number of Attributes:	0
Object Ref:	516409

Dataspace and Datatype

No. of Dimension(s):	3
Dimension Size(s):	32 x 256 x 256
Max Dimension Size(s):	32 x 256 x 256
Data Type:	8-bit unsigned integer

Storage Layout:	CHUNKED: 16 X 256 X 256
Compression:	4.098:1GZIP: level = 2
Filters:	GZIP
Storage:	SIZE: 511785, allocation time: Incremental
Fill value:	NONE

Figure 2. View of an IMS file in HDFView.

[2] The point of view that the HDF5 library is the lowest level of the IMS format definition has served us well for more than 15 years. The library across its different versions has encapsulated any binary format modifications so well that we have generally not taken much notice thereof.

3 A Minimal IMS File

To understand the minimum format requirements we will start by describing all those parts of an IMS file that must be present for the Imaris software to read the file. What is minimal has slightly changed between Imaris versions. Here we describe what is a minimal IMS file for Imaris version 9.6.
The IMS format requires the root group of the HDF5 file to have a subgroup DataSet within which the image data and histograms thereof are contained and another subgroup called DataSetInfo within which parameters are contained.

3.1 “DataSet” Group

The DataSet group must contain a subgroup “ResolutionLevel 0” which must contain a subgroup “TimePoint 0” which must contain a subgroup “Channel 0” which must contain an HDF5 Dataset called “Data” and an HDF5 dataset called “histogram”. The “ResolutionLevel 0” subgroup contains all high resolution image data for all time points and all channels. When the image isn’t tiny the DataSet group must contain several “ResolutionLevel X” subgroups containing downsampled versions of the high resolution image data. The requirements for these Resolution levels will be described in the section on ResolutionLevel Requirements.

The meaning of the “TimePoint X” subgroups in the “ResolutionLevel 0” group is self-explanatory. Each “TimePoint X” group must contain the image and histogram data for the corresponding time point of the image.

3.1 “Channel X” Group

The “Channel X” subgroups in the “TimePoint X” groups must contain the image and histogram data for the corresponding channel at the given time point. The “Channel X” group must also contain a “Data” dataset and a “histogram” dataset and it must have the following HDF5 attributes:

Attribute Name	Attribute Value (stored as String Array)
ImageSizeX	Image Size in X
ImageSizeY	Image Size in Y
ImageSizeY	Image Size in Z
HistogramMax	Max Intensity in “Data” of “Channel X”
HistogramMin	Min Intensity in “Data” of “Channel X”

The ImageSize attributes are needed because the dataset “Data” has a size that is necessarily a multiple of the chunk size. The ImageSize thus cannot be deduced from the dimensions of the “Data” dataset and must be stored separately in the ImageSize attributes of the “Channel X” group. Similarly the min and max of the histogram range are not part of the histogram dataset but stored as attributes in the “Channel X” group.

3.1.1. The “Data” Dataset

The HDF5 dataset “Data” in the “Channel X” group contains the image data for the corresponding resolution level, time point and channel. This dataset is a 3-dimensional scalar HDF5 dataset of data type 16-bit unsigned integer or 8-bit unsigned integer.

For optimal performance the dataset should have chunks that correspond to the blocks used by Imaris for rendering and analysis. For performance reasons Imaris does not (!) adapt its blocksize to the chunk size in an HDF5 file. Imaris reads data with chunk sizes different from Imaris blocksize too but this has a negative performance impact.

The data stored in “Data” may be compressed using gzip compression. Since version 9.6 of Imaris they may also be compressed with LZ4 compression and the data may be “shuffled” to store consecutively low precision bytes and high precision bytes of 16bit data.

3.1.1. The “Histogram” Dataset

The HDF5 dataset “histogram” in the “Channel X” group contains the histogram of the image data stored in the corresponding “Data” dataset. This histogram must be a dataset of datatype 64-bit unsigned integer and it must have 256 bins of equal size in the range given by the HistogramMax and HistogramMin attributes of the corresponding “Channel X” group.

3.1.2. The “Histogram1024” Dataset

For 16 bit images the HDF5 dataset “Histogram1024” in the “Channel X” group contains a histogram of the image data stored in the corresponding “Data” dataset that is constructed with 1024 bins. This histogram must be a dataset of datatype 64-bit unsigned integer and it must have 1024 bins of equal size in the range given by the HistogramMax1024 and HistogramMin1024 attributes of the corresponding “Channel X” group. For 8 bit images such a histogram is not necessary.

3.2 DataSetInfo Group

The DataSetInfo group must be present within the top level group and it must contain some minimal information about the image. It must contain a subgroup image that specifies the geometry of the image in terms of the minimum and maximum coordinates of its bounding box.

3.1.1. The “Image” Group

The “Image” group within the DataSetInfo group contains attributes describing the image geometry.

Attribute Name	Attribute Value (stored as String Array)
ExtMin0	X-Coordinate for the minimum of the image bounding box.
ExtMin1	Y-Coordinate for the minimum of the image bounding box.
ExtMin2	Z-Coordinate for the minimum of the image bounding box.
ExtMax0	X-Coordinate for the maximum of the image bounding box.
ExtMax1	Y-Coordinate for the maximum of the image bounding box.

ExtMax2	Z-Coordinate for the maximum of the image bounding box.

Note that the minimum and maximum of the bounding box lie on the border of the border voxels, not in the center of the border voxels. This means that an image consisting of a single slice in Z will have ExtMax2 > ExtMin2..

4 A Minimal IMS File with Color Information

The minimal IMS file described above does not have color information for the channels. To add color information for rendering image data in a specific color the DataSetInfo group must have a subgroup "Channel X" the attributes of which specify the rendering colors.

4.1.1. The "DataSetInfo/Channel X" Group

The "Channel X" groups within the DataSetInfo group contain attributes describing the channel color.

Attribute Name	Attribute Value (stored as String Array)
Color	RGB color. Each color is in the range 0-1 and stored as string with 3 decimals. The three colors are separated by a whitespace.
ColorMode	Possible values: "BaseColor", "ColorTable"
ColorOpacity	Opacity value in the range 0-1 stored as string with 3 decimals.
ColorRange	Min and Max intensity values for the color range. Intensities below Min are rendered black by Imaris. Intensities above Max are rendered with maximum brightness.

5 A Multiresolution IMS File

A typical IMS file contains multiple resolution levels for the purpose of high performance rendering. A minimal multi-resolution IMS file contains the information of the minimal IMS file described above and additionally contains more "ResolutionLevel X" subgroups in the DataSet group.Each "ResolutionLevel X" group is organized in the same way as the "ResolutionLevel 0" group described in the minimal IMS file section. The size of the low resolution images must be computed according to the following method.

5.1 Size of Low Resolution Images

Starting from the highest resolution the lower resolution image sizes generally result from dividing the high resolution image size by two, and, if the result is not an integer, rounding to the next higher integer. There is one exception: the size of the low resolution image is for some dimension X, Y, or Z kept identical to the high resolution image size if the size of this dimension is particularly small in comparison to the others. The following pseudocode gives the exact recipe used by Imaris to determine when the size should be reduced along each of the dimensions.

```
UInt64 vLargeSizeX = vLargeImage->GetSizeX();
UInt64 vLargeSizeY = vLargeImage->GetSizeY();
UInt64 vLargeSizeZ = vLargeImage->GetSizeZ();
bool vReduceX = (10*vLargeSizeX)*(10*vLargeSizeX) > vLargeSizeY*vLargeSizeZ;
bool vReduceY = (10*vLargeSizeY)*(10*vLargeSizeY) > vLargeSizeX*vLargeSizeZ;
bool vReduceZ = (10*vLargeSizeZ)*(10*vLargeSizeZ) > vLargeSizeX*vLargeSizeY;
UInt64 vSmallSizeX = vReduceX ? vLargeSizeX/2 : vLargeSizeX;
UInt64 vSmallSizeY = vReduceY ? vLargeSizeY/2 : vLargeSizeY;
UInt64 vSmallSizeZ = vReduceZ ? vLargeSizeZ/2 : vLargeSizeZ;
```

Two examples of multi-resolution pyramids consistent with the IMS format are given in Table 1.

Level	SizeX	SizeY	SizeZ
0	7643	5246	1552
1	3821	2632	776
2	1910	1316	388
3	955	658	194
4	477	329	97
5	238	164	48

Level	SizeX	SizeY	SizeZ
0	34664	22043	23
1	17332	11021	23
2	8666	5510	23
3	4333	2755	23
4	2166	1377	23
5	1083	688	23
6	541	344	23
7	270	172	23

Table 1. Two examples of multi-resolution pyramids of the IMS.

5.2 Number of Resolution Levels

The number of resolution levels that an IMS file should contain is determined by the method of section 5.1 to compute the size of the lower resolution images up to the point where the number of voxels decreases below 4*1024*1024 = 4194304. At this admittedly somewhat arbitrary point further downsampling to smaller resolution levels wouldn't produce much benefit for rendering or analysis and thus only one resolution level with less than 4194304 voxels is created.

5.3 Binning Image Data

To create the low resolution images the image data of the higher resolution images have to be resampled into the lower resolution images. This is done in the simplest way by binning. Each low resolution voxel is computed from 8 or 4 higher resolution voxels by averaging the intensities of the higher resolution voxels and rounding the average to the next higher integer value.

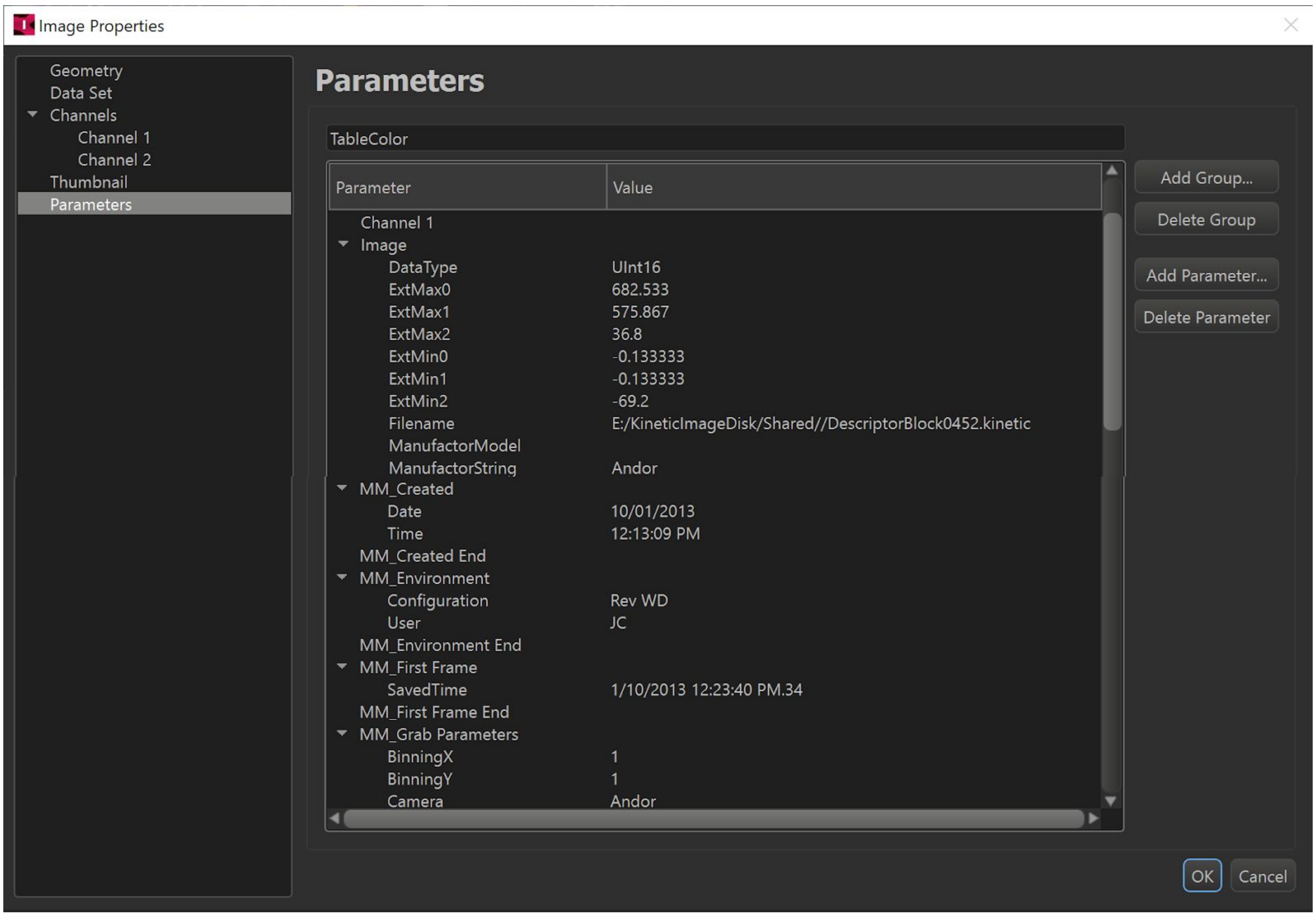
6 A TimeSeries IMS File

From the description of the minimal IMS file it might already be clear how time series image data will be stored in the “DataSet” group. Naturally in this case the “Resolution Level X” groups contain subgroups “TimePoint 0” … “TimePoint N” each of which contains the image data for one time point at the respective resolution level.
With time series comes the need for precise information about the time and duration of each time point. This information is stored within a subgroup of “DataSetInfo” called “TimeInfo”. The “TimeInfo” group contains one “TimePointX” attribute storing the time of that time point in the format YYYY-MM-DD HH:MM:SS.SSS.

Attribute Name	Attribute Value (stored as String Array)
DataSetTimePoints	The number of time points in the DataSet
FileTimePoints	The number of time points in the file (currently the same as DataSetTimePoints)
TimePointX	Time for time point X in the format YYYY-MM-DD HH:MM:SS.SSS, e.g. 1991-10-01 16:45:45.000 . Time must be strictly increasing between time points.

7 Optional Information in DataSetInfo

The DataSetInfo group may contain more information than what was described above. In terms of what Imaris needs to function this additional information is optional but is stored in the file to provide information to the user of the file. This information can be accessed in Imaris within the Parameters section of the Image Properties dialog shown in Figure 3.



Parameters

TableColor

Parameter	Value
Channel 1	
Image	
DataType	UInt16
ExtMax0	682.533
ExtMax1	575.867
ExtMax2	36.8
ExtMin0	-0.133333
ExtMin1	-0.133333
ExtMin2	-69.2
Filename	E:/KineticImageDisk/Shared//DescriptorBlock0452.kinetic
ManufactorModel	
ManufactorString	Andor
MM_Created	
Date	10/01/2013
Time	12:13:09 PM
MM_Created End	
MM_Environment	
Configuration	Rev WD
User	JC
MM_Environment End	
MM_First Frame	
SavedTime	1/10/2013 12:23:40 PM.34
MM_First Frame End	
MM_Grab Parameters	
BinningX	1
BinningY	1
Camera	Andor

Figure 3. The ImageProperties dialog of Imaris shows all data stored in first level subgroups of "DataSetInfo".

7.1 Useful "DataSetInfo/Channel X" informatoin

Optional information about each channel that may be of interest to a user includes the following:

Attribute	Value
LSMEmissionWavelength	Emission Wavelength
LSMExcitationWavelength	Excitation Wavelength
LSMPhotons	Deconvolution parameter
LSMPinhole	Pinhole diameter
MicroscopeMode	Deconvolution parameter
Name	Short description of the channel (some chars)
NumericalAperture	Numerical Aperture
Offset	Deconvolution parameter

Pinhole	Deconvolution parameter
RefractionIndexEmbedding	Deconvolution parameter
RefractionIndexImmersion	Deconvolution parameter
Gain	Deconvolution parameter

7.2 “DataSetInfo/Image”

Optional information about the image that may be of interest to a user includes the following:

Attribute	Value
LensPower	Deconvolution parameter
Name	Short description of the image
Noc	Number of channels
RecordingDate	“YYYY-MM-DD HH:MM:SS”
Unit	Unit of coordinates: "m“, "mm“, "um“ or "nm“

7.2 Other Subgroups of DataSetInfo

Within the “DataSetInfo” group it is possible to create subgroups with attributes. Any number of subgroups or attributes within each subgroup may be created. This feature is used to pass on metadata about the image or the acquisition from the original file format.