

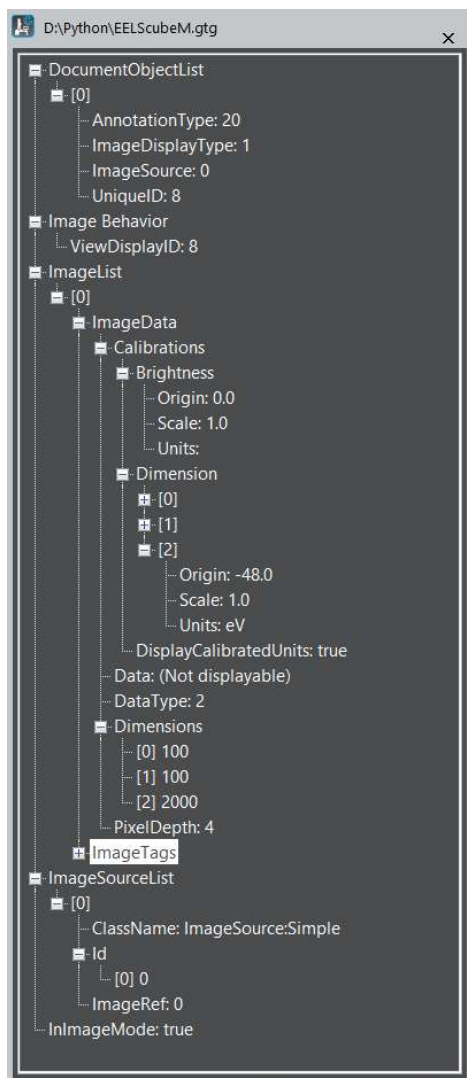
Notes on DigitalMicrograph dm3 / dm4 formats

Pavel Potapov, 2022

Composed with a kind help from Bernhard Schaffer.

These notes refine and extend previous notes by Greg Jefferis and Chris Boothroyd.

The important difference between the dm3 and dm4 formats is that the former uses uint32 for storing the lengths of data fragments, which is limited to 2GB, e.g. the images exceeding 2G cannot be stored. The dm4 format uses uint64 with no practical limitations on size. In this notes, these two variants are denoted as 4/8 bytes, uint32/uint64. The former should be used for the dm3 format and the latter for the dm4 one.



The major body of a file consists of several tag groups defining how the image information is interpreted and displayed in DigitalMicrograph. According Bernhard Schaffer, only few of these tag groups (shown in the fig.) are obligatory for reading a file. The rest are set to default when first opening the file.

'ImageDisplay' tag should be 1 for images and 3 for line profiles.

The image information is placed in the 'ImageList' group. Several images can be stored in one file, for instance, line profiles along certain line, image insets et cet. Those images are enumerated in the 'ImageList' group as [0],[1],[2]... At least one image is obligatory. All files I saw consist also of a small thumbnail image, which is however not obligatory. Inside each such enumerated tag group, there are subgroups 'ImageTags' and 'ImageData'. The former consists of metadata (not obligatory) and the latter consists of data itself. Then, in the 'ImageData' group there are 'Calibrations' subgroup (not obligatory), 'Dimensions' (define the image size in all dimensions, obligatory) subgroups and 'Data' (image as an array) tag. Also, tags 'DataType' and 'PixelDepth' are obligatory there. They point to the type of elements in the image array (integer, float et cet.) and the number of bytes for each element.

Tag 'ImageRef' indicates which image in the 'ImageList' group should be displayed.

How these tag groups are stored?

DigitalMicrograph binary files start with

a **file header**:

- *4 bytes, uint32* **3 or 4** depending of whether it is the dm3 or dm4 format. It seems that DigitalMicrograph determines the kind of the format from this note, not from the file extension.
- *4/8 bytes, uint32/uint64* length of all tag groups. It does not include the file header and the last 8 zero bytes.
- *4 bytes, uint32* byte order in tag notes. **0**: big endian, **1**: little endian (standardly used in Windows). This should not be confused with the endian of the headers notes, which are always in big endian.

Each tag group starts with

a **group header**:

- *8 bytes, uint64* Only in dm4: length of the group in bytes starting from the end of the note. For some reasons, this note is absent in the groups of the root folder but presents in all subgroups.
- *1 byte, uint8* **1** or **0** indicating whether the group is sorted or not. The exact meaning of this parameter is not fully clear. Assumedly, this means that group is (or is not) sorted alphabetically. This parameter is important: when it is set incorrectly, file opening might fail.
- *1 byte, uint8* **1** or **0** indicating whether the group should be open or not when browsing the tag structure. This parameter seems to be unimportant.
- *4/8 bytes, uint32/uint64* number of tags in the group.

Then a **name header** follows:

- *1 byte, uint8* **20** or **21**. The code **20** shows that a tag group is expected, the code **21** shows that this is an individual tag.
- *4 bytes, uint32* length of the tagGroup/tag label in *n* bytes.
- *(1 byte, uint8) n times* Label of the tagGroup/tag in the 'utf8' or 'latin-1' coding. Some tagGroup/tag might have no names and the length of their labels is correspondingly zero. When reading, DigitalMicrograph enumerates such tags as '0', '1', '2'...

In case of a tag group (code **20**), a new group header follows and this may continue infinitely in a matryoshka manner. If an individual tag is expected (code **21**), we find next

A **tag header**:

- *8 bytes, uint64* Only in dm4: length of the tag in bytes starting from the end of the note.
- *4 bytes, uint32* A marker '%%%' denoting the beginning of the tag.
- *4/8 bytes, uint32/uint64* length definition *m*. This shows how many numbers is needed to describe the tag type.

- In a simplest case of a singular number, this is **1**: just one code is needed to describe the type of a number.
- In case of an array tag, we need **3** parameters to describe it: the first parameter showing that this is an array (code **20**), the second parameter is an array size and the third parameter defines the type of array elements.
- In case of a tuple of **n** elements, the length definition equals **3+2*n**. This is because the type of a tuple and the types of each individual element need to be described. For instance, for a tuple consisting of two float elements, this would be *uint32/uint64=15 (code of tuple), uint32/uint64=0 (just delimiter), uint32/uint64=2 (number of elements in the tuple), then twice [uint32/uint64=0 (delimiter), uint32/uint64=6 (code of float32 type)]; totally 7 notes.*

It is possible that more structures exist, but I was not able to figure it out.

- (4/8 bytes, uint32/uint64) *m* times numbers defining the type of a tag

Then the tag itself follows. This could be just one number consisting of several bytes as defined by the type. Or, this could be a sequence of bytes forming an array according its type definition.

Finally, each dm3/dm4 file ends up with 8 zero bytes. This is an obligatory note and a file missing this might fail in opening.

Coding of types:

Type codes in DigitalMicrograph files:

2	short	int16
3	long	int32
4	unsigned short	uint16
5	unsigned long	uint32
6	float	float32
7	double	float64
8	bool	uint8
9	char	uint8
11	long long	int64
12	unsigned long long	uint64
15	structure	tuple
18	string	
20	array	

Type codes of DigitalMicrograph images:

1	short	int16
2	float	float32
3	complex	complex64
6	char	uint8
7	long	int32
9	byte	int8
10	unsigned short	uint16
11	unsigned long	uint32
12	double	float64
13	complex	complex128
14	bool	uint8
23	thumbnail	

One should distinguish the type codes used for reading/writing dm3/dm4 files and the types of images used, for instance, in DigitalMicrograph scripting.