Femtonics ADVANCED MICROSCOPY

Importing .mesc file contents into MATLAB®

Abstract

This document describes a preliminary process of importing data into MATLAB® from .mesc files generated by the MESc application from Femtonics Ltd.

Background

.mesc files generated by the MESc application are valid HDF5 files and can be opened, inspected and read with any HDF5 tool. You can get all the information content out of .mesc files by using any of the available HDF5 tools.

It is to be noted, however, that the MESc application makes strict assumptions about the internal organization of .mesc files. Modifying a .mesc file with general HDF5 tools can therefore easily prevent the file from being opened by the MESc application and is therefore not recommended. Recovery from such situations is not supported by Femtonics Ltd. in any way. To be safe, we recommend to always open .mesc files in a read-only way with HDF5 tools, or to always work on a copy of your .mesc files, or have good backups.

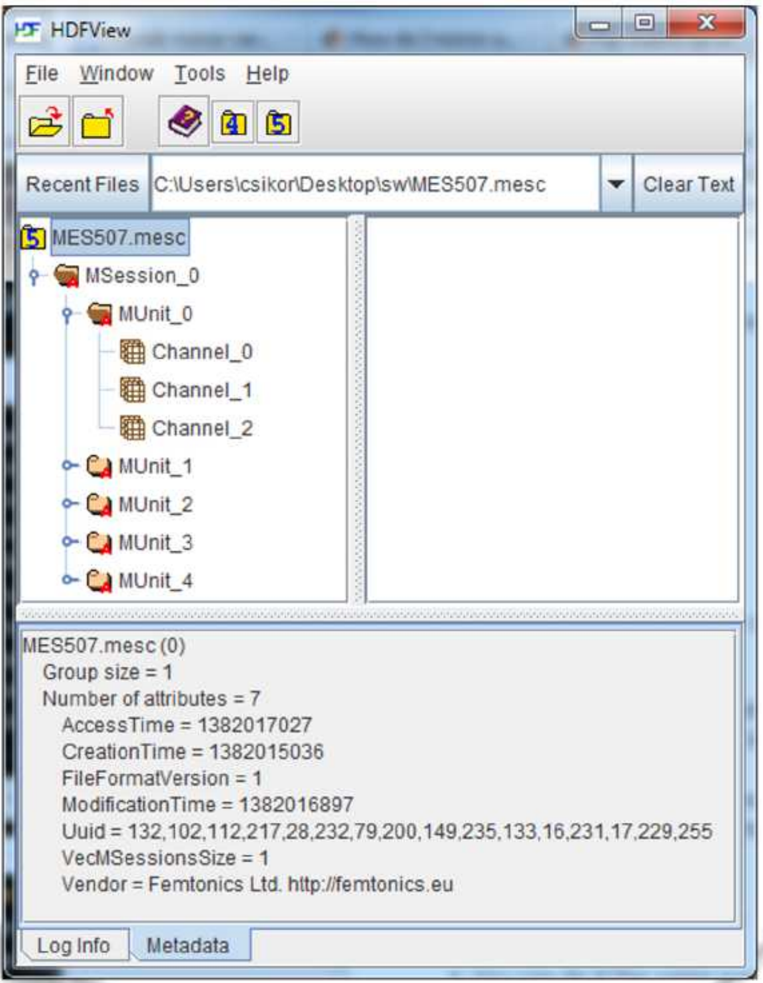
Please also note that the .mesc file format is actively developed and will regularly change in the future. The present document refers only to version 1 of the .mesc file format used in version 1.0 of the MESc application and will become obsolete at some point in time.

To solve the above two issues, Femtonics Ltd. is developing a MATLAB® MEX extension for handling .mesc files on the MESc abstraction level. This tool will be able to read and modify .mesc files and will follow future file format changes, too. This guide is only relevant until this tool is finished.

Please note that Femtonics Ltd. guarantees full backward compatibility for .mesc files in both the MESc application and the .mesc file handling MATLAB® MEX extension under development. That is, you will always be able to open .mesc files generated by older software versions with newer software versions.

Getting an overview of the .mesc file structure with HDFVIEW

The easiest way to introduce yourself to the structure of .mesc files is the free HDFVIEW program from The HDF Group. You can see a screenshot below.



```
MES507.mesc (0)
  Group size = 1
  Number of attributes = 7
    AccessTime = 1382017027
    CreationTime = 1382015036
    FileFormatVersion = 1
    ModificationTime = 1382016897
    Uuid = 132,102,112,217,28,232,79,200,149,235,133,16,231,17,229,255
    VecMSessionsSize = 1
    Vendor = Femtonics Ltd. http://femtonics.eu
```

The tree view to the center left shows a portion of the contents of the file "MES507.mesc". The root group "/" of the file contains a HDF5 group "/ MSession_0" representing a measurement session. The HDF5 group "/ MSession_0" contains several HDF5 groups named "/MSession_0/MUnit_#" representing measurement units. Measurement units contain HDF5 datasets "/ MSession_0/MUnit_#/Channel_#" representing the data contents of channels.

The text field to the bottom of the screenshot shows the attributes of the root group "/" in the file "MES507.mesc" (please note that the root group is highlighted in the tree view). The .mesc file format stores lots of useful information in the attributes of HDF5 groups and datasets. Many of them are self-explanatory, some are not that obvious. This document offers you only some introductory information on the semantics of these attributes. The MATLAB® MEX-based solution mentioned above will offer you as complete access to .mesc file contents as possible.

Root group attributes

This section explains all attributes of .mesc root groups ("/") in detail (see the HDFVIEW screenshot above). Most .mesc attributes can be understood based on this information. When in doubt, please contact Femtonics Ltd. with your questions.

AccessTime, CreationTime, and ModificationTime

These timestamps are refreshed on .mesc file access, creation, and modification, respectively. These and all other timestamps are stored in the Unix (POSIX) time format, i.e., they represent the seconds elapsed since 00:00:00 1970-01-01, measured in Coordinated Universal Time (UTC). Sometimes, an additional numeric field is added to provide sub-second time resolution (usually nanoseconds).

Converting these timestamps to the format preferred by your application is a one or two step process. First, you need to convert the Unix time to your preferred date-time format. The second, optional step is to convert the result of the first step from UTC to your local time. Most software environments provide tools to achieve both of these steps. A MATLAB® specific solution is presented later in this document.

Vendor

String fields such as Vendor are stored in one of two different formats. For ASCII strings, a one-dimensional array of 8-bit numbers (characters) is used. UNICODE strings are stored in one-dimensional arrays of 16-bit numbers (wide characters). A MATLAB® specific code example is presented later in this document.

Uuid

Each MESc object in the .mesc file has a randomly generated Universally unique identifier for easy programmatic identification. It is always stored in the Uuid field.

FileFormatVersion

Numeric attributes such as FileFormatVersion are always stored in the most appropriate numeric data type. These can be directly read by MATLAB®.

VecMSessionsSize

Attributes such as VecMSessionsSize denote the size of the smallest vector able to store a collection of MESc objects (in this case, measurement sessions). In the screenshot, VecMSessionsSize=1 because there is only one measurement session in the root group of the opened file. Equivalently, valid measurement session indices are in the left-closed, right-open interval [0,1[. A size parameter of N would denote an index interval [0,N[. Note, however, that not all indices in this interval are necessarily valid. When an object is deleted from a MESc object vector, the indices of the other MESc objects in the vector are not updated. In addition, new objects are always appended to the end of object vectors. These mechanisms altogether achieve perpetual index stability within each .mesc file.

Datasets

The .mesc file format uses HDF5 datasets to store the data contents of channels. HDF5 datasets can be directly read by any standard HDF5 tool, e.g. MATLAB®. There are, however, two caveats.

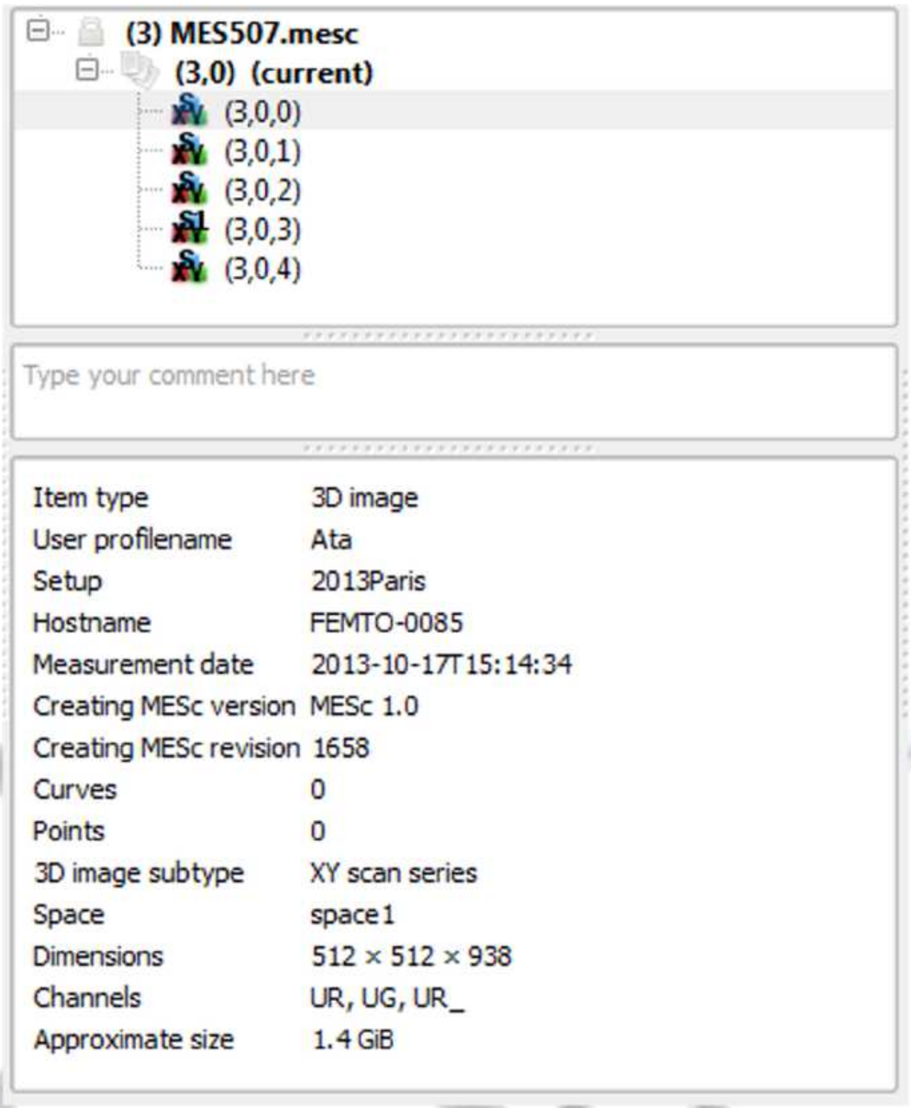
First, HDF5 and MATLAB® use the opposite matrix indexing convention for multidimensional arrays. Therefore, you need to permute matrix indices after channel data import.

Second, .mesc stores measurement results in raw format, i.e., it stores the exact numeric values that arrived from the microscope hardware to the measurement control PC. Raw data arrays are always accompanied by a conversion object which can transform raw values into physically meaningful units (and back). There are, however, several different kinds of conversion objects and describing them in detail would make the present description overly long. Instead, the specific conversion used for resonant scan data is given below, together with a MATLAB® code example.

A MATLAB® code example for dealing with these two technical details is presented later in this document.

.mesc file presentation in MESc

The part of the file MES507.mesc shown in the HDFVIEW screenshot above is displayed by the MESc application in the way below.



(3) MES507.mesc
(3,0) (current)
(3,0,0)
(3,0,1)
(3,0,2)
(3,0,3)
(3,0,4)

Type your comment here

Item type	3D image
User profilename	Ata
Setup	2013Paris
Hostname	FEMTO-0085
Measurement date	2013-10-17T15:14:34
Creating MESc version	MESc 1.0
Creating MESc revision	1658
Curves	0
Points	0
3D image subtype	XY scan series
Space	space1
Dimensions	512 × 512 × 938
Channels	UR, UG, UR_
Approximate size	1.4 GiB

Note that measurement session indices and measurement unit indices are indicated in this view, too. E.g., the highlighted measurement unit has the index vector (3,0,0) which denotes "opened file 3, measurement session 0, measurement unit 0". Looking at these indices might help you identifying which parts of the .mesc file to process with MATLAB® or some other HDF5-capable data processing software.

Extracting information from .mesc files with MATLAB®

As discussed above, .mesc files are valid HDF5 files; therefore, they can be handled with the built-in HDF5 tools of MATLAB®. It is important to stress again that changing .mesc files in this way might make the MESc application refuse to open the file. Therefore, read-only .mesc file access is recommended at this stage, as is shown in the code examples below.

The MESc application generates .mesc files with version 1.8 of the HDF5 library, while MATLAB® version R2007b shipped with Femtonics microscopes uses version 1.6. Unfortunately, this prevents MATLAB® version R2007b from importing a few bits and pieces of .mesc files, but, luckily, all important information is still accessible. Our MATLAB® MEX solution in development for .mesc file handling will solve this problem, too, by including the exact same HDF5 library version used by MESc.

MATLAB® offers both a high-level and a low-level HDF5 handling capability. Probably due to the HDF5 library version incompatibility mentioned above, the high-level HDF5 API in MATLAB® version R2007b cannot open .mesc files. The low-level HDF5 API, however, mostly works. You can get an introduction to its usage in the MATLAB® help under the heading Programming → Working with Scientific Data Formats → Hierarchical Data Format (HDF5) Files → Using the MATLAB Low-Level HDF5 Functions.

MATLAB® code example 1: listing all HDF5 attributes from a HDF5 group or dataset

The following listing is a valid MATLAB® function with line numbers prepended to each line of code for easy reference. You should have received a version of this code without the line numbers, too, attached to this document under the name listMEScH5ObjAttribs.m. Copying this attachment to a file named listMEScH5ObjAttribs.m somewhere in your MATLAB® path makes this function accessible from your MATLAB® session. We explain some important details below the code example.

```
function listMEScH5ObjAttribs(path, objname, tzoffset)
% LISTMESCH5OBJATTRIBS lists the attributes of a HDF5 object in a MESc file.
%  listMEScH5ObjAttribs( path, objname, tzoffset ) lists
%  the names, classes, and values of each HDF5 attribute
%  in the HDF5 group or dataset OBJNAME in the HDF5 file at PATH.
%  Optional argument TZOFFSET denotes the hour offset of the local
%  timezone relative to UTC (e.g., 1 for Central European Time and 2 for
%  Central European Summer Time). The default value of TZOFFSET is 0.

if nargin < 2
   error('usage: listMEScH5ObjAttribs(path, objname[, tzoffset])')
end

if ~exist('tzoffset', 'var')
   tzoffset=0; % set default tzoffset
end

% open HDF5 file for reading
try
   fileID=H5F.open(path, 'H5F_ACC_RDONLY', 'H5P_DEFAULT');
catch
   error('Unable to open file ''%s''.', path)
end

% open HDF5 group or dataset
try
   objID=H5G.open(fileID, objname);
catch
   try
      objID=H5D.open(fileID, objname);
   catch
      error('Unable to open HDF5 object ''%s''.', objname)
   end
end

% iterate over all contained attributes
numAttrs=H5A.get_num_attrs(objID);
for attrIdx=0:numAttrs-1
   try
      % read data from the currently inspected attribute
      attribID=H5A.open_idx(objID, attrIdx);
      attribName=H5A.get_name(attribID);
      attribValue=H5A.read(attribID, 'H5ML_DEFAULT');
      attribClass=class(attribValue);
      H5A.close(attribID);

      % print attribute name, class, and value
      disp(['--> attribute ' num2str(attrIdx) ': name: ''' attribName ...
         ''', class: ' attribClass ', value:' ])
      if ischar(attribValue)
         % pretty-print 8-bit string
         fprintf('%s\n\n',attribValue')
      elseif strcmp(attribClass, 'int16')
         % pretty-print 16-bit string
         fprintf('%s\n\n',char(attribValue'))
      elseif strcmp(attribClass, 'uint64') && ...
            ( ~isempty(regexp(attribName, 'Time$', 'once')) ...
            || ~isempty(regexp(attribName, 'DatePosix$', 'once')))
         % pretty-print date to 1 second precision
         fprintf('%s\n\n',datestr(datenum(1970, 1, 1, 0, 0, ...
            double(attribValue))+tzoffset/24))
      else
         % print any other type of data
         disp(attribValue')
      end
   catch
      % issue warning for unreadable attribute (due to some HDF5
      % forward incompatibility)
      fprintf('--> attribute %d: ***unreadable***\n\n', attrIdx)
   end
end

% HDF5 objects and files are closed automatically on function exit
```

A possible invocation of the above function is listMEScH5ObjAttribs ('MES507.mesc','/',2) which should give you the same pieces of information as the attributes seen in the HDFVIEW screenshot above, but in a processed form. For instance, time attributes are displayed as date strings and are shown for Central European Summer Time (UTC+2). (Unfortunately, MATLAB® version R2007b provides no means to automatically retrieve the time zone used by the host computer.)

The code is mostly self explanatory, but let me point you out a couple of interesting spots.

1. Line 20 demonstrates how to open a .mesc file in read-only mode to prevent file modification and a possible loss of .mesc file structure.

2. Lines 27 and 30 show that different functions should be used to open HDF5 groups and datasets.

3. Lines 37 and 38 give you an example of iterating through all attributes of a HDF5 object by attribute index. Attribute name based attribute handling is demonstrated in the next code example.

4. Line 64 demonstrates that, due to the opposite MATLAB® and HDF5 conventions for multidimensional matrix indexing, imported two-dimensional matrices should be transposed (and the indices of imported multidimensional matrices should be reversed).

5. Lines 52, 55, and 60 demonstrate the handling of ASCII and UNICODE strings, and Unix time values, respectively.

6. Line 69 refers to the case when a HDF5 attribute cannot be imported due to the HDF5 library version incompatibility mentioned above.

MATLAB® code example 2: reading a frame from a resonant scan movie recorded by MESc

The following listing is a valid MATLAB® function with line numbers prepended to each line of code for easy reference. You should have received a version of this code without the line numbers, too, attached to this document under the name readMEScMovieFrame.m. Copying this attachment to a file named readMEScMovieFrame.m somewhere in your MATLAB® path makes this function accessible from your MATLAB® session. You can find some explanation below the code example.

```matlab
function data = readMEScMovieFrame(...
    path, msessionIdx, munitIdx, channelIdx, frameIdx)
% READMESCMOVIEFRAME returns a frame from a movie in a .mesc file.
%   readMEScMovieFrame( path, msessionIdx, munitIdx, channelIdx, frameIdx )
%   returns a frame from a .mesc movie measurement unit.
%   Parameters:
%     PATH        .mesc file path
%     MSESSIONIDX measurement session index (usually 0)
%     MUNITIDX    measurement unit index (indexed from 0)
%     CHANNELIDX  channel index (indexed from 0)
%     FRAMEIDX    index of the frame to return (indexed from 0)

if nargin < 5
    error('usage: data = readMEScMovieFrame(path, msessionIdx, munitIdx, channelIdx, frameIdx)')
end

% open HDF5 file for reading
try
    fileID=H5F.open(path, 'H5F_ACC_RDONLY', 'H5P_DEFAULT');
catch
    error('Unable to open file "%s".', path)
end

% open the HDF5 group representing the measurement unit
groupname=sprintf('/MSession_%d/MUnit_%d', msessionIdx, munitIdx);
try
    groupID=H5G.open(fileID, groupname);
catch
    error('Unable to open measurement unit.')
end

% open the HDF5 dataset holding the channel contents
try
    datasetID=H5D.open(fileID, ...
        sprintf('%s/Channel_%d', groupname, channelIdx));
catch
    error('Unable to open channel.')
end

% get the necessary attributes
attribID=H5A.open_name(groupID, 'XDim');
xdim=H5A.read(attribID, 'H5ML_DEFAULT');
H5A.close(attribID);
attribID=H5A.open_name(groupID, 'YDim');
ydim=H5A.read(attribID, 'H5ML_DEFAULT');
H5A.close(attribID);
attribID=H5A.open_name(groupID, 'ZDim');
zdim=H5A.read(attribID, 'H5ML_DEFAULT');
H5A.close(attribID);
if frameIdx >= zdim
    error('Section index is too large; section does not exist.')
end

start = [frameIdx 0 0];
stride = [1 1 1];
count = [1 1 1];
block = [1 ydim xdim];
dataspaceID = H5D.get_space(datasetID);
H5S.select_hyperslab(dataspaceID,'H5S_SELECT_SET',start,stride,count, block);
dataspaceID_memory = H5S.create_simple(3,block,[]);

data = 65535 - permute( ...
    H5D.read(datasetID, 'H5ML_DEFAULT', dataspaceID_memory, ...
    dataspaceID, 'H5P_DEFAULT'),[2 1]);

% HDF5 objects and files are closed automatically on function exit
```

A possible invocation of the above function would be imagesc (flipud (readMEScMovieFrame('MES507.mesc',0,0,0,101))). This should pop up a plot window showing the 101th frame from the 0th channel of the 0th measurement unit of the 0th measurement session in the file MES507.mesc and should refer to the corresponding plot in the View panel of MESc (albeit with a different color look-up table). In the command, imagesc does the plotting, and flipud flips the 2D matrix upside down to bridge the different axis conventions in MESc and MATLAB®.

Again, the code is mostly self explanatory, but let me point you out a couple of interesting spots.

1. Lines 41 to 49 demonstrate attribute name based attribute handling.

2. Lines 54 to 60 shows you how to select a frame from a three-dimensional HDF5 dataset. This is often necessary when a resonant scan movie does not fit into your computer's RAM. Of course, you are free to open any cuboid-shaped portion of HDF5 datasets, or even the whole dataset. Note that, contrary to MATLAB®, everything in the HDF5 library is indexed from 0.

3. Lines 62 to 64 demonstrate both multidimensional index reversal with the permute function and performing the linear data conversion used for all resonant scan measurements, $x \rightarrow 65535 - x$.