DRAFT

2007 December 11

The OME-HDF Specification

Purpose
This white paper is to outline the specification of the OME-HDF format. We will discuss the reasoning behind the creation of such a format, and outline the format itself.

Why a new file format?
In the life sciences community, there are a wide variety of different file formats. Dozens of formats exist for light microscopy alone, not to mention the plethora of them relating to electron microscopy, high-throughput screening, tomography, medical imaging, and other areas. Why add another format?

The current batch of microscopy formats are mostly designed around the concept of image planes. There is typically some quantity of image slices, stored as (X, Y) co-ordinate data, and other dimensions (Z, T, C, etc) are stored as "rasterizations" of the XY planes. This approach can be cumbersome in higher dimensions, and the ability to store data in more complex dimensional structures is becoming increasingly important as scientists create datasets with more and more features and higher dimensionality. For instance, at the Laboratory for Optical and Computational Instrumentation (LOCI), scientists are beginning to gather spectral lifetime data in seven dimensions—length, width, depth, time, emission spectra wavelengths, lifetime histograms, and cell polarity—which is not explicitly supported in any current file format, including OME-XML or OME-TIFF. Other possible dimensions such as excitation wavelengths will probably be explored in the future as well. OME-TIFF cannot rasterize arbitrary dimensions in an arbitrary order, including orders that are ideal for spectral lifetime, such as putting lifetime bins sequentially.

In addition, TIFF in general is also not suited for non-planar data. Storing this kind of data in TIFF can be misleading to the user. Generally, TIFF programs will display image "planes" to the user. In many types of multi-dimensional data, displaying the data this way (e.g., each lifetime histogram bin as its own "plane") is meaningless, where users expect displayed images to have meaning.

Another reason for a new format is the problem of data compression. In current formats, compression—if it is done at all—is usually done either per plane (e.g., LZW-TIFF or JPEG) or across the entire file (e.g., ZIP). Compression per plane does not exploit possible similarity between image planes, whereas compressing the entire file generally forces the programmer to decompress the entire file in memory or on disk, which is at the very least computationally expensive, if not resource intensive as well. Compression is becoming more and more important as scientists are gathering and transferring larger and larger datasets in more dimensions. An ideal data format would allow compression across different dimensions or sets of dimensions arbitrarily. For example, video codecs take advantage of not only compression within a single image, but across images as well, to exploit similarity over time.

However, there are several things that are important to carry over from existing formats, such as ability to store metadata. Without in-file metadata, it is difficult to carry non-pixel information along with the dataset. At best, an outside file is kept storing information such as researcher, microscope information, etc. At worst, this data is lost altogether.

It is important for these and other reasons that a new file format be able to support, at minimum, the following:

- Arbitrary dimensionality
- Arbitrary file size
- Compression over arbitrary parts of the image data, and possibly non-similarly sized/dimensioned parts of image data
- Metadata storage

The OME-HDF format

To meet these design goals, we merge two existing formats, OME-XML and HDF.

It is important to note that this document does not constitute a full formal specification of the OME-HDF format. This is meant as an overview of the format, and a discussion of the issues that are involved in creating the format. If this format were to undergo further development, a full specification would likely be a good first step.

OME-XML (http://www.ome-xml.org/) is well known in the microscopy community as a data format, and has the ability to store arbitrary metadata, with a schema developed for most common microscope and experiment acquisition metadata. OME-XML provides a strong basis for our metadata needs.

As an aside, we wish to note that the OME-XML schema has been developed as a flexible data model for use in a variety of contexts. While we feel that the majority of people's needs can be met with the four main OME-XML-based formats—OME-HDF, OME-TIFF, OME-XML with embedded base64-encoded pixels, and OME-XML as a companion metadata file—we do not want to create the impression that the intended uses are limited to those. The OME consortium is supportive of alternative uses of OME-XML in situations where the existing approaches are insufficient for some reason.

HDF (http://hdf.ncsa.uiuc.edu/) is well known in the space science community for its ability to store multiple-dimensioned data of arbitrary size. HDF is the primary data format for NASA's Earth Observing System. Over the fifteen-year lifespan of the project, they expect to store fifteen petabytes of data using HDF. Additionally, HDF is being supported by an increasing number of data platforms such as NetCDF 4. HDF is also gaining popularity throughout various scientific communities; in microscopy, version 5.5 of Bitplane's Imaris file format uses HDF as its basis. It could be beneficial to the biological community to harness the power of this format.

OME-HDF combines these two formats by encoding OME-XML metadata within an HDF data structure.

The OME-HDF Specification
OME-HDF uses groups and normal datasets to store its data. Below is an example of the tree structure of an OME-HDF file, with further detail following.

```
+ / (root)
|-+ xml_root
| |-+ 1e_OME
|   |-+ 2a_xmlns
|   |  \- http://www.openmicroscopy.org/Schemas/OME/2007-06
|   |-+ 3a_xmlns:CA
|   |  \- http://www.openmicroscopy.org/Schemas/CA/2007-06
|   |-+ 4c
|   |  \- example
|   ...
|-+ pixels_root
  |-+ urn:lsid:example.com:Pixels:01
    |-+ 5d_p:2,c:16,t:16,z:1,y:240,x:320
    |  \- 42
    |-+ 6p_p:0
    | |-+ 7p_c:0
    | | |-+ 8p_t:0
    | |   |-+ 9p_z:0
    | |     \- (a 2d dataset containing x and y)
    | |-+ 10p_c:1
    |   |-+ 11p_t:0
    |     |-+ 12c_LZW
    |       \- (a 3d dataset containing x, y, and z, LZW compressed)
    |-+ 13p_p:1
...
```

The root group of an HDF file contains two groups: xml_root, and pixels_root.

xml_root
xml_root is the parent group of a representation of an OME-XML metadata block. The group tree under xml_root is constructed in such a way as to be able to recreate the original OME-XML block, or to be able to scan through the tree to look for the necessary information. There will only be one xml_root in an OME-HDF file.

XML in general contains three types of items of concern: elements, attributes, and character data.

Elements (such as <OME>) are added to the HDF as groups. Anything that is inside an element, such as character data or other elements, will be found as a child of the element. Since group names must be unique, elements are prepended with a unique (within the file) numeric identifier, followed by "e_", to indicate that this group represents an element, followed by the name of the element (e.g., 1e_OME).

Attributes are also added to the HDF as a group. An attribute will have only one child, which will be a String dataset, containing the value of the attribute. Attributes are prepended with a unique numeric identifier, followed by an "a_" to indicate that this is an attribute, followed by the text of the attribute (e.g., 2a_xmlns).

Character data is also identified as a group, which is a numeric identifier followed by "c_". The actual character data will be represented in the lone child of this group as a String dataset.

pixels_root
pixels_root is the parent group of any sets of pixels available in the OME-HDF file. There will be only one pixels_root in any OME-HDF file.

All children of the pixels_root will be groups with unique names matching the regex /(urn:lsid:)?(\S+\.\S+)+(:Pixels)(:\S+)/. An example of such a name is "urn:lsid:example.com:Pixels:01". All such children represent the roots of their individual pixel sets.

Each pixels set will contain a dimensions group, and a number of top-level pixel elements.

The dimensions group will be named with a unique numeric identifier followed by a "d_". This will be followed by a string containing colon-separated pairs, themselves separated by commas. The pairs will be of the form "dimension_name:dimension_size." So, for example, "p:2" indicates that the p dimension has size 2. These pairs will be ordered with the fastest-rasterized dimension coming last.

The dimensions group will also contain a single 1-dimensional data item of the type used by the pixels for this pixels set. The number used (42 in the example above) is unimportant.

The Pixels group above (the urn:lsid:example.com:Pixels:01 group, for example) also contains any top level pixel nodes. Consider a multi-dimensional dataset as a tree. At the top, you have parent nodes of the "slowest moving" raster order, in this case p. As children of each parent node, you have the next slowest moving, in this case c. This can continue all the way down to the bottom of the tree.

Each pixel group node consists of a unique numeric identifier, followed by "p_", followed by the name of the dimension, followed by ":", and the number of this particular dimension index.

At the leaf nodes, the data for each node is contained as an n-dimensional dataset of the correct data type, corresponding to one of the basic data types that HDF supports (see http://www.hdfgroup.uiuc.edu/HDF5/doc_1.8pre/doc/H5.format.html#DatatypeMessage for more information).

While a leaf node may contain a single data value, this case is not usually what will occur. At any point in the hierarchy, if a dataset leaf node is found, it is assumed to contain the rest of the data that would be expected under that tree. In the example, after p:0, c:0, t:0, and z:0, a dataset leaf node would contain all of the xy data for these dimensional indices.

Compression
Instead of such a leaf node, a compression node may be specified. Such a group node will be named containing a unique numeric identifier, followed by "c_" and the type of compression that will be used. Before this format is finalized, a number of compression types should be specified, although a program reading this format should feel free to reject any compression type with which it is not familiar.

If a dataset is compressed, it will be stored as a one-dimensional dataset of type uint8, representing the compressed byte stream. The uncompressed byte stream is expected to be in the raster order of the remaining dimensions, and of size and type matching the rest of the file. Each compression type will unambiguously identify the algorithm for converting the n-dimensional dataset of the original data type into a one-dimensional byte stream, and vice versa, including parameter values and resolution of endianness issues.

A compression node may contain additional pixel group nodes, rather than a single dataset leaf node. In this case, the given type of compression is considered default for all dataset leaf nodes under the group.

Other topics
Again, this is not a full formal specification of an OME-HDF format. There are a number of issues that should be addressed as part of such a specification, including but not limited to the following:

- How should a reader react to missing data (such as missing pixel data, non-specified compression, missing dimensionality data, dimensions that are present that were not specified, etc.)
- How are "dimension-pixels" numbered? zero to n-1? 1 to n? Arbitrarily, assuming ascending order?
- What kinds of compression are allowed, how are they implemented, etc.
- Any possible limitations of the HDF format (imposed by restrictions of HDF or OME-XML)
- Handling of extra data not discussed in the spec.
- Possible addition of further branches for later format expansion

Conclusion
With OME-HDF we have attempted to create a format that supports arbitrary dimensionality, compression of arbitrary chunks of datasets, with the ability to store microscope and experiment metadata, on datasets of arbitrary size. We believe that this format meets all of these goals, but any comments or suggestions to improve the format are welcome.

It is important to note that this document does not constitute a full formal specification of the OME-HDF format. It is meant as an overview of the format, and a discussion of the issues that are involved in creating the format. As OME-HDF undergoes further development, a full specification will be forthcoming.