OME Data Model and File Formats

The Open Microscopy Environment

May 10, 2023

CONTENTS

This documentation covers the OME Data Model, OME-XML and OME-TIFF.

The OME Model is a specification for storing data on biological imaging. The model includes image parameters, such as XYZ dimensions and pixels type, as well as extensive metadata on, for example, image acquisition, annotation, and regions of interest (ROIs). This common specification is essential for the exchange of image data between different software packages. OME-XML is a file format used to store data according to the OME Model, serving as a convenient file format for data migration from one site or user to another. OME-TIFF is a multi-plane tiff file that contains OME metadata in the header, in the form of OME-XML. This allows the pixels to be read with any TIFF-compatible program, and the metadata to be extracted with any OME-aware application. Our paper describing the design and implementation of the OME-XML file appeared in Genome Biology.

The OME consortium currently provides three major tools capable of working with OME-XML and OME-TIFF:

- The Bio-Formats library is a full-featured library with many features related to OME-XML, including conversion of third party file format metadata into OME-XML structures. It can write image data to the OME-TIFF format.
- OME Files C++ is a reference implementation of the OME data model and OME-TIFF for C++ developers wishing to support these in their own software. It can read and write OME-TIFF data.
- The OMERO server works directly with OME-XML. It can import data from OME-XML and OME-TIFF, as well as export to OME-TIFF.

If you have used OME-XML, OME-TIFF, Bio-Formats or OMERO in your work please use the correct citations to acknowledge us.

We have received support from several companies who use our file formats, for details see our list of Partners.

Note: The versioning for this documentation reflects updates to any of the components it covers and is therefore incremented more frequently than the Schema version, which only covers the Data Model and for which a version history is provided below.

CHAPTER

ONE

OME-TIFF

1.1 The OME-TIFF format

1.1.1 OME-TIFF file structure

The following section discusses various considerations when designing OME-TIFF filesets distributed over multiple files.

Single versus multiple files

Splitting a fileset across multiple files can have advantages in terms of acquisition but also processing purposes. The OME-TIFF file format can support any image organization. However, using one TIFF file per timepoint per channel with the focal planes for that timepoint and channel stored sequentially within the TIFF makes for very easy creation of *TiffData* elements (see *5D datasets*).

The main downside of splitting OME-TIFF over multiple files is their inherent fragility to common file-system operations such as file renaming or file copying which have the potential to "break" the fileset.

See *OME-TIFF sample data* for examples of single OME-TIFF file versus OME-TIFF distributed over multiple files.

File size

The TIFF file format internally uses 32-bit byte offsets. The largest offset which can be represented is 4GB, making this value the upper limit of the file size supported by the design.

The OME-TIFF file format supports the BigTIFF file extensions allowing the use of 64-bit byte offsets to overcome this size limit.

The main limitation of the BigTIFF file extension is the degree of its adoption in the community. Although OME Bio-Formats and OMERO will handle OME-TIFF using the BigTIFF file extension, other tools might not be able to open it.

Metadata redundancy

Storing multiple planes per OME-TIFF file and keeping the OME-XML metadata embedded in every file header is recommended (see the *5D datasets*) but is not always practical. Normally, the OME-XML metadata block is small in comparison with the binary pixel data in the file, but in some cases, it may be disproportionately larger.

Common reasons for this situation include:

- storing each image plane in its own TIFF file
- having a large amount of metadata such as plane-specific timestamps
- having many structured annotations in the OME-XML.

For example, if you have a dataset with 1,000 time points, with each plane recorded at 512 × 512 as uint8 pixel type, storing each plane in its own file uncompressed requires ~256KB of disk per file, and ~256MB total. But if you have 5MB of corresponding OME-XML metadata, embedding a copy of that metadata in every file would result in a dataset nearly 20 times larger than before, requiring ~5.5MB of disk per file, and more than 5GB total.

One of the advantages of reproducing the OME-XML metadata across files is redundancy. If one of the constituent files survives data loss, the metadata survives. The space tradeoff of this duplication is acceptable when compared to the bulk of the pixel data in most cases, provided a suitable number of planes are stored in each TIFF.

Companion file vs master OME-TIFF file

Since the *2011-06 version* of the OME-XML schema, it is possible to store partial OME-XML metadata blocks in some or all of the TIFF files pointing to a master file containing the full OME-XML metadata (see the *technical specification* for more details).

The master file can be either a master OME-TIFF file or a companion OME-XML file (see the *Multi-file OME-TIFF filesets*).

Using a companion OME-XML file allows information that can only be generated at the end of the acquisition to be easily appended without the need to manipulate existing TIFF IFDs. It has the same drawbacks as those highlighted above in the sense that this companion file needs to be preserved as part of the fileset to prevent metadata loss.

See also:

Proposed tweak to µManager data files in 2.0
Community discussion about usage of companion file in OME-TIFF filesets

1.1.2 OME-TIFF specification

The following provides technical details on the OME-TIFF format. It assumes familiarity with both the TIFF specification and the OME Data Model, although there is some review of both.

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this specification are to be interpreted as described in RFC 2119.

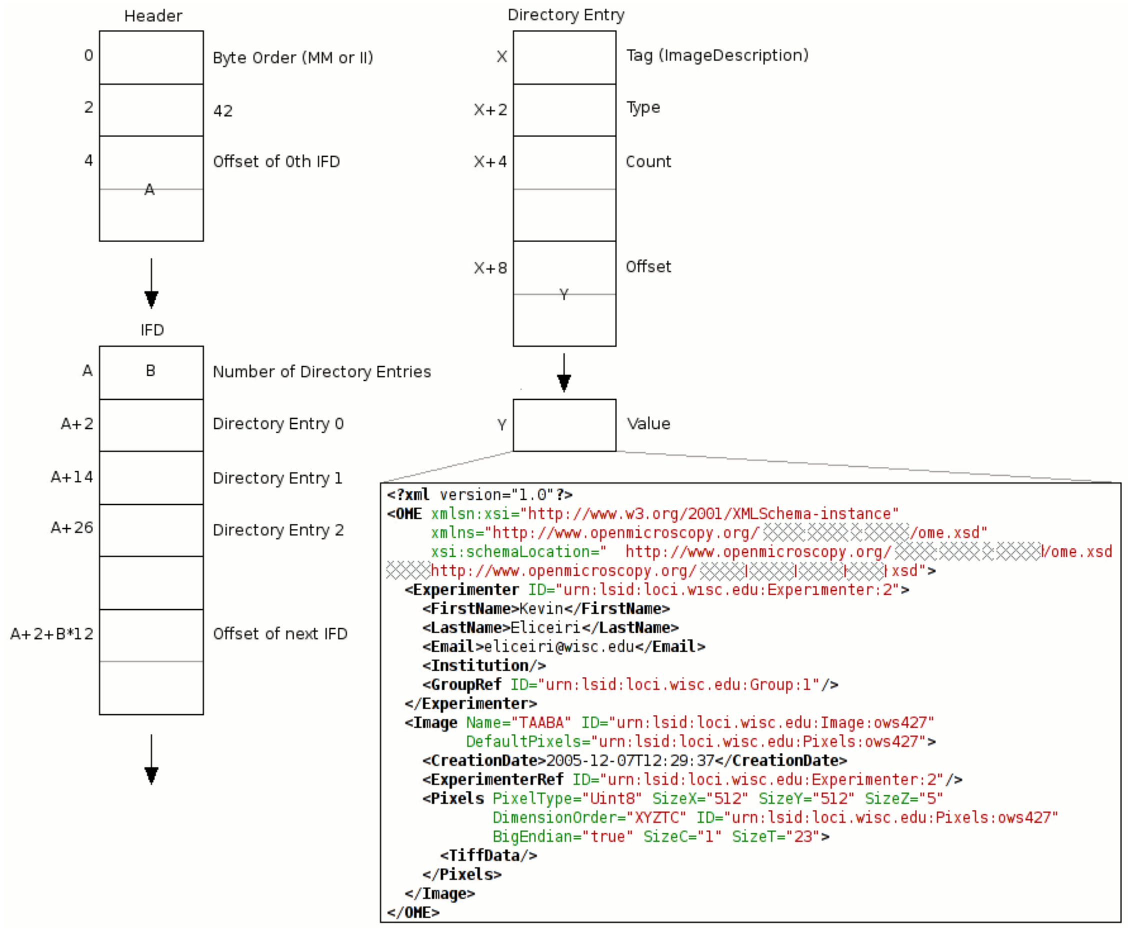
An OME-TIFF dataset consists of:

- one or more files in standard TIFF format with the file extension `.ome.tif` or `.ome.tiff` or BigTIFF format with one of these same file extensions or a BigTIFF-specific extension `.ome.tf2`, `.ome.tf8` or `.ome.btf`
- a string of OME-XML metadata embedded in the ImageDescription tag of the first IFD (Image File Directory) of each file. The XML string must be UTF-8 encoded.

Note that BigTIFF-specific file extensions are an addition to the original specification, and software using an older version of the specification may not be able to handle these file extensions.

OME-XML metadata

This string is a standard block of OME-XML, except that instead of storing pixels as BinData elements with base64-encoded pixel data beneath them, it references pixels with TiffData elements that specify which TIFF IFD corresponds to each image plane. As such, the OME-XML string can be regarded as a "metadata block" because the actual pixels are stored within the TIFF structure, not the XML.



```
<?xml version="1.0"?>
<OME xmlsn:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="http://www.openmicroscopy.org/[illegible]/ome.xsd"
     xsi:schemaLocation="  http://www.openmicroscopy.org/[illegible]/ome.xsd
[illegible]http://www.openmicroscopy.org/[illegible]xsd">
  <Experimenter ID="urn:lsid:loci.wisc.edu:Experimenter:2">
    <FirstName>Kevin</FirstName>
    <LastName>Eliceiri</LastName>
    <Email>eliceiri@wisc.edu</Email>
    <Institution/>
    <GroupRef ID="urn:lsid:loci.wisc.edu:Group:1"/>
  </Experimenter>
  <Image Name="TAABA" ID="urn:lsid:loci.wisc.edu:Image:ows427"
         DefaultPixels="urn:lsid:loci.wisc.edu:Pixels:ows427">
    <CreationDate>2005-12-07T12:29:37</CreationDate>
    <ExperimenterRef ID="urn:lsid:loci.wisc.edu:Experimenter:2"/>
    <Pixels PixelType="Uint8" SizeX="512" SizeY="512" SizeZ="5"
            DimensionOrder="XYZTC" ID="urn:lsid:loci.wisc.edu:Pixels:ows427"
            BigEndian="true" SizeC="1" SizeT="23">
      <TiffData/>
    </Pixels>
  </Image>
</OME>
```

Fig. 1: OME-TIFF header

The diagram *OME-TIFF header* (adapted from the TIFF specification) shows the organization of a TIFF header along with the placement of the OME-XML metadata block. Note this is for the TIFF standard specification only; the header structure is slightly different for BigTIFF; see the BigTIFF file format specification. A TIFF file can contain any number of IFDs, with each one specifying an image plane along with certain accompanying metadata such as pixel dimensions, physical dimensions, bit depth, color table, etc. One of the fields an IFD can contain is ImageDescription, which provides a place to write a comment describing the corresponding image plane. This field is a convenient place to store the OME-XML metadata block—any TIFF library capable of parsing IFDs and extracting an ImageDescription comment can easily obtain an OME-TIFF file's entire set of metadata as OME-XML.

Note: A TIFF file contains one IFD per image plane, but the OME-XML metadata block is stored only in the first IFD structure. However, for an image sequence distributed across multiple OME-TIFF files, each file will contain a copy of the OME-XML metadata block (see *Partial OME-XML metadata* below for exceptions to this rule). Thus, if any files are lost, the metadata is preserved. The OME-XML block in each file is nearly identical—the only difference is in the TiffData elements appearing beneath Pixels elements, since each TIFF file contains a different portion of the image data (see *Multi-file OME-TIFF* below).

DimensionOrder

As mentioned above, the standard OME-XML format encodes image planes as base64 character blocks contained within BinData elements beneath a Pixels element. The Pixels element has a DimensionOrder attribute that specifies the rasterization order of the image planes. For example, XYZTC means that there is a series of image planes with the Z axis varying fastest, followed by T, followed by C (e.g. if a XYZTC dataset contains two focal planes, three time points and two channels, the order would be: Z0-T0-C0, Z1-T0-C0, Z0-T1-C0, Z1-T1-C0, Z0-T2-C0, Z1-T2-C0, Z0-T0-C1, Z1-T0-C1, Z0-T1-C1, Z1-T1-C1, Z0-T2-C1, Z1-T2-C1).

Since a multi-page TIFF has no limit to the number of image planes it can contain, the same scheme described above for specifying the rasterization order works within the OME-TIFF file. The only difference is that instead of the pixels being encoded in base64 inside BinData elements, they are stored within the TIFF file in the standard fashion, one per IFD; see the *TiffData element* below for specifics.

TIFF comments

When embedding your OME-XML string as a TIFF comment, it is best practice to preface it with the following informative comment:

```
<!-- Warning: this comment is an OME-XML metadata block, which contains
crucial dimensional parameters and other important metadata. Please edit
cautiously (if at all), and back up the original data before doing so.
For more information, see the OME-TIFF documentation:
https://docs.openmicroscopy.org/latest/ome-model/ome-tiff/ -->
```

The TiffData element

As the illustration *OME-TIFF header* shows, all that is needed to indicate that the pixels are located within the enclosing TIFF structure is a TiffData element with no attributes. By default, the first IFD corresponds to the first image plane (Z0-T0-C0), and the rasterization order of subsequent IFDs is given by the Pixels element's DimensionOrder attribute, as described above.

However, there are several attributes for TiffData elements allowing greater control over the dimensional position of each IFD:

- IFD - gives the IFD(s) for which this element is applicable. Indexed from 0. Default is 0 (the first IFD).
- FirstZ - gives the Z position of the image plane at the specified IFD. Indexed from 0. Default is 0 (the first Z position).
- FirstT - gives the T position of the image plane at the specified IFD. Indexed from 0. Default is 0 (the first T position).
- FirstC - gives the C position of the image plane at the specified IFD. Indexed from 0. Default is 0 (the first C position).
- PlaneCount - gives the number of IFDs affected. Dimension order of IFDs is given by the enclosing Pixels element's DimensionOrder attribute. Default is the number of IFDs in the TIFF file, unless an IFD is specified, in which case the default is 1.

Here are some example XML fragments:

Fragment 1

```
<Pixels ID="urn:lsid:loci.wisc.edu:Pixels:ows325"
        Type="uint8" DimensionOrder="XYZTC"
        SizeX="512" SizeY="512" SizeZ="3" SizeT="2" SizeC="2">
    <TiffData/>
</Pixels>
```

IFD	Position
0	Z0-T0-C0
1	Z1-T0-C0
2	Z2-T0-C0
3	Z0-T1-C0
4	Z1-T1-C0
5	Z2-T1-C0
6	Z0-T0-C1
7	Z1-T0-C1
8	Z2-T0-C1
9	Z0-T1-C1
10	Z1-T1-C1
11	Z2-T1-C1

The default TiffData tag simply assigns every IFD to a position according to the given DimensionOrder rasterization. If the TIFF file has more than SizeZ*SizeT*SizeC (3*2*2=12 in this case) IFDs, the remaining IFDs are extraneous and should be ignored by OME-TIFF readers.

Fragment 2

```
<Pixels ID="urn:lsid:loci.wisc.edu:Pixels:ows462"
        Type="uint8" DimensionOrder="XYCTZ"
        SizeX="512" SizeY="512" SizeZ="4" SizeT="3" SizeC="2">
    <TiffData PlaneCount="10"/>
</Pixels>
```

IFD	Position
0	Z0-T0-C0
1	Z0-T0-C1
2	Z0-T1-C0
3	Z0-T1-C1
4	Z0-T2-C0
5	Z0-T2-C1
6	Z1-T0-C0
7	Z1-T0-C1
8	Z1-T1-C0
9	Z1-T1-C1

When specified, the PlaneCount attribute gives the number of IFDs affected by the TiffData element. In this case, even though the Pixels element defines 4*3*2=24 image planes total, the TiffData element assigns only 10 planes. The remaining 14 planes are unspecified and hence lost.

Fragment 3

```
<Pixels ID="urn:lsid:loci.wisc.edu:Pixels:ows197"
        Type="uint8" DimensionOrder="XYZTC"
        SizeX="512" SizeY="512" SizeZ="4" SizeC="3" SizeT="2">
    <TiffData IFD="3" PlaneCount="5"/>
</Pixels>
```

IFD	Position
3	Z0-T0-C0
4	Z1-T0-C0
5	Z2-T0-C0
6	Z3-T0-C0
7	Z0-T1-C0

States that the rasterization begins at the fourth IFD (IFD #3), and continues for five planes total. IFDs #0, #1 and #2 are not used, and should be ignored by OME-TIFF readers.

Fragment 4

```
 <Pixels ID="urn:lsid:loci.wisc.edu:Pixels:ows789"
         Type="uint8" DimensionOrder="XYZTC"
         SizeX="512" SizeY="512" SizeZ="1" SizeC="1" SizeT="6">
    <TiffData IFD="0" FirstT="5"/>
    <TiffData IFD="1" FirstT="4"/>
    <TiffData IFD="2" FirstT="3"/>
    <TiffData IFD="3" FirstT="2"/>
    <TiffData IFD="4" FirstT="1"/>
    <TiffData IFD="5" FirstT="0"/>
</Pixels>
```

IFD	Position
0	Z0-T5-C0
1	Z0-T4-C0
2	Z0-T3-C0
3	Z0-T2-C0
4	Z0-T1-C0
5	Z0-T0-C0

Any number of TiffData elements may be provided. In this case, the dimensional positions of each of the first six IFDs are explicitly defined, effectively overriding the rasterization given by DimensionOrder, storing the planes in reverse temporal order.

For details on validating your OME-XML metadata block, see the validating OME-XML section on the *Extracting, processing and validating OME-XML* page.

Multi-file OME-TIFF

As demonstrated above, the OME-TIFF format is capable of storing the entire multidimensional image series in one master OME-TIFF file.

Alternatively, a collection of multiple OME-TIFF files may be used. Using the TiffData attributes outlined above together with UUID elements and attributes, the OME-XML metadata block can unambiguously define which dimensional positions correspond to which IFDs from which files. Each OME-TIFF need not contain the same number of images.

The only difference between the OME-XML metadata block per TIFF file is the UUID attribute of the root OME element. This value should be a distinct UUID value for each file, so that each TiffData element can unambiguously reference its relevant file using a UUID child element.

Note: The FileName attribute of the UUID is optional, but strongly recommended—otherwise, the OME-TIFF reader must scan OME-TIFF files in the working directory looking for matching UUID signatures.

When splitting an OME-TIFF across multiple files, the OME-XML metadata must either be embedded into each TIFF file or use partial metadata blocks.

Embedded OME-XML metadata

In the first case, a nearly identical OME-XML metadata block must be inserted into the first IFD of each constituent OME-TIFF file.

The only difference between the OME-XML metadata block per TIFF file is the UUID attribute of the root OME element. This value should be a distinct UUID value for each file, so that each TiffData element can unambiguously reference its relevant file using a UUID child element.

The *5D datasets* demonstrate how OME-TIFF datasets can be distributed across multiple files. Each of the files in the set has identical metadata apart from the *UUID*, the unique identifier in each file. For example the identifiers could be distributed as follows:

tubhiswt_C0_TP0.ome.tif with ID 45c8bf18-6aa2-478c-9080-e0b0d3eb1f70

```
<OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     Creator="OME Bio-Formats 5.2.0-m4"
     UUID="urn:uuid:45c8bf18-6aa2-478c-9080-e0b0d3eb1f70"
     xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06
     http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd">
...
    <Pixels BigEndian="false" DimensionOrder="XYZTC" ID="Pixels:0"
            Interleaved="false" SignificantBits="8" SizeC="2" SizeT="43"
            SizeX="512" SizeY="512" SizeZ="10" Type="uint8">
...
      <TiffData FirstC="0" FirstT="0" FirstZ="0" IFD="0" PlaneCount="1">
        <UUID FileName="tubhiswt_C0_TP0.ome.tif">
          urn:uuid:45c8bf18-6aa2-478c-9080-e0b0d3eb1f70
        </UUID>
      </TiffData>
...
      <TiffData FirstC="0" FirstT="1" FirstZ="0" IFD="0" PlaneCount="1">
        <UUID FileName="tubhiswt_C0_TP1.ome.tif">
```

(continues on next page)

(continued from previous page)

```
          urn:uuid:743275b7-6726-46bd-b7bb-aca3085f429a
        </UUID>
      </TiffData>
...
```

tubhiswt_C0_TP1.ome.tif with ID 743275b7-6726-46bd-b7bb-aca3085f429a

```
<OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     Creator="OME Bio-Formats 5.2.0-m4"
     UUID="urn:uuid:743275b7-6726-46bd-b7bb-aca3085f429a"
     xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06
     http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd"
...
    <Pixels BigEndian="false" DimensionOrder="XYZTC" ID="Pixels:0"
            Interleaved="false" SignificantBits="8" SizeC="2" SizeT="43"
            SizeX="512" SizeY="512" SizeZ="10" Type="uint8">
...
      <TiffData FirstC="0" FirstT="0" FirstZ="0" IFD="0" PlaneCount="1">
        <UUID FileName="tubhiswt_C0_TP0.ome.tif">
          urn:uuid:45c8bf18-6aa2-478c-9080-e0b0d3eb1f70
        </UUID>
      </TiffData>
...
      <TiffData FirstC="0" FirstT="1" FirstZ="0" IFD="0" PlaneCount="1">
        <UUID FileName="tubhiswt_C0_TP1.ome.tif">
          urn:uuid:743275b7-6726-46bd-b7bb-aca3085f429a
        </UUID>
      </TiffData>
...
```

And so on for files:

- **tubhiswt_C0_TP2.ome.tif** with ID 1f462b60-b508-446e-b42a-c4e6fa2a44e8
- **tubhiswt_C0_TP3.ome.tif** with ID a023901d-43fd-44f2-b4be-159afa1e985c
- …

Partial OME-XML metadata

Instead of embedding the full OME-XML metadata into the header of each OME-TIFF, partial OME-XML metadata blocks can be stored in some or all of the OME-TIFF files using the Binary-Only element as illustrated below:

```
<?xml version="1.0" encoding="UTF-8"?>
<OME UUID="urn:uuid:4978087c-a670-4b12-af53-256c62d8d101"
     xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06
     http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd">
   <BinaryOnly MetadataFile="multifile.companion.ome"
               UUID="urn:uuid:07504f88-7bc3-11e0-b937-2faf67bc00b3"/>
</OME>
```

The MetadataFile element should contain the name of the master file containing the full OME-XML metadata block and UUID should contain the UUID of this master file.

The master file containing the full OME-XML metadata should be either an OME-XML companion file with the extension `.companion.ome` or a master OME-TIFF file containing the full metadata (see *Multi-file OME-TIFF filesets* for representative samples).

Sub-resolutions

New in version 6.0.0.

OME-TIFF supports multi-resolution images or pyramidal images where individual planes are stored at different levels of resolution. The downsampled image planes are called pyramidal levels, sub-resolution image planes or sub-resolutions.

Supported resolutions

OME-TIFF planes can be reduced along the X and Y dimensions. Each pyramidal level must be a downsampling of the full-resolution plane in the X and Y dimensions and the resolution should stay unchanged in the other dimensions. The downsampling factor:

- should be an integer value,
- should be identical along the X and the Y dimensions,
- should stay the same between each consecutive pyramid level.

The following table below shows two examples of pyramid level dimensions using typical downsampling factors:

	Example 1 (X × Y × Z × C × T)	Example 2 (X × Y × Z × C × T)
Downsampling factor	3	4
Level 0 (full-resolution)	9234 × 6075 × 1 × 1 × 10	38912 × 25600 × 200 × 3 × 1
Level 1	3078 × 2025 × 1 × 1 × 10	9728 × 6400 × 200 × 3 × 1
Level 2	1026 × 675 × 1 × 1 × 10	2432 × 1600 × 200 × 3 × 1
Level 3	342 × 225 × 1 × 1 × 10	608 × 400 × 200 × 3 × 1
Level 4	114 × 74 × 1 × 1 × 10	152 × 100 × 200 × 3 × 1
Level 5	38 × 25 × 1 × 1 × 10	

Storage

Full-resolution image planes must remain stored as described above using a valid TIFF IFD and referenced from the OME-XML metadata using the *TiffData* element.

Each sub-resolution must be stored in the same IFD as the full-resolution image plane. Additionally:

- the offsets of all sub-resolutions IFDs must be referenced from the IFD of the full-resolution plane using the SubIFDs TIFF extension tag 330 as defined in the TIFF Tech Note 1 of the Adobe PageMaker® 6.0 TIFF Technical Notes. The list of sub-resolution offsets must be ordered by plane size from largest to smallest,
- the IFD offsets of pyramidal levels must neither be referenced in the primary chain of IFDs derived from the first IFD of the TIFF file nor be referenced in a *TiffData* element of the OME-XML metadata,
- the NewSubFileType TIFF tag 254 for each pyramidal level should be set to 1 to distinguish full-resolution planes from downsampled planes.

The planes of largest resolutions should be organized into tiles rather than strips as described in the TIFF specification and may be compressed using any of the officially supported schemes including LZW, JPEG or JPEG2000. Sub-resolution image planes may choose to use different compression algorithms than the one used by the full resolution plane. For example the full resolution image may use no compression or lossless compression while the sub-resolution images use lossy compression.

While baseline TIFF may suffice for smaller pyramidal images, BigTIFF is recommended for large images.

See also:

https://openmicroscopy.github.io/design/OME005/
Official design proposal for the addition of sub-resolution support to the OME-TIFF specification.

1.1.3 OME-TIFF example source code for common operations

This section discusses some common operations related to the OME-TIFF format, and provides example source code in Java for performing them. We strongly recommend you read and understand the *OME-TIFF specification* before attempting to deploy any of the following code.

- Extracting a TIFF comment – demonstrates how to extract the OME-XML comment from an OME-TIFF file.
- Modifying a TIFF comment – demonstrates how to modify an OME-XML comment within an OME-TIFF file.
- Converting other formats to OME-TIFF – demonstrates how to convert third-party formats to OME-TIFF.

The Bio-Formats library provides a lot of functionality related to OME-XML and OME-TIFF, to ease the burden of file format handling and conversions. Rather than offer source code that performs all of the above actions on its own, we instead offer examples that utilize Bio-Formats, for more robust and succinct operation.

Extracting a TIFF comment

To extract a comment from a TIFF file without the aid of a TIFF library, the following steps are required:

1. Read in the 8-byte header.
2. Determine if the file is a valid TIFF, and if so, whether its byte order is big endian or little endian. Bytes 0–1 must equal “II” (0x4949, little endian, “Intel”) or “MM” (0x4D4D, big endian, “Motorola”).
3. Check TIFF version. Bytes 2–3 must equal 42 (0x2A) or 43 (0x2B) with the proper endianness, which are the TIFF specification or the newer BigTIFF specification, respectively.
4. If the TIFF version is 0x2A, offsets are 4 bytes (32-bit). If the TIFF version is 0x2B, bytes 4–5 are the size of offsets in bytes (should be 0x0008 for 8-byte 64-bit offsets, but could be different), and bytes 6-7 are padding (should be 0x0000).
5. Determine the byte offset into the file of the first IFD. For TIFF version 0x2A, this information is stored in bytes 4–7, with the proper endianness. For TIFF version 0x2B, this information is stored starting at byte 8, sized according to the specified offset size and with the proper endianness. This will be bytes 8–15 for 8-byte offsets.
6. Skip to the first IFD, and read in the IFD’s header.
7. Iterate through the directory entries looking for the ImageDescription (270) tag.
8. Jump to the offset given by the ImageDescription entry, and read the number of bytes indicated by the entry.
9. Convert the bytes to an ASCII string.

The Bio-Formats command line tools include a program, tiffcomment, that performs these steps using the getComment(String) method of TiffParser. You can produce a nicely formatted OME-XML string from an OME-TIFF file with:

```
tiffcomment file.ome.tif | xmlindent
```

See also:

BigTIFF file format specification

Modifying a TIFF comment

Modifying a TIFF comment can be tricky because the length of the altered OME-XML string is unlikely to be the same as before. As such, the IFD's ImageDescription directory entry must be updated to reflect the new byte count. In addition, if the string is longer than before, it will no longer fit at its old offset, unless the comment was at the end of the file, so the entry's offset might need to change as well.

The `TiffSaver.overwriteIFDValue()` method within Bio-Formats, efficiently alters a directory entry with a minimum of waste. The count field of the entry is intelligently updated to match the new length. If the new length is longer than the old length, it appends the new data to the end of the file and updates the offset field; if not, or if the old data is already at the end of the file, it overwrites the old data in place.

The following program extracts comments from TIFF files, prompts the user to alter the comments on the command line, and writes updated comments back to the files. It requires the Bio-Formats library.

EditTiffComment.java

The comment string is acquired using `new TiffParser(f).getComment()`, and updated with

```
TiffSaver saver = new TiffSaver(f);
RandomAccessInputStream in = new RandomAccessInputStream(f);
saver.overwriteComment(in, xml);
```

To quickly edit an OME-TIFF files comment on the command line use `tiffcomment -edit filename.ome.tiff` from the Bio-Formats command line tools.

Converting other formats to OME-TIFF

One of the major goals of Bio-Formats is to standardize the metadata from all supported third-party formats into OME-XML. Doing so makes conversion to OME-TIFF very straightforward—just write the pixels to TIFF however you want (e.g. with libtiff), and store the converted OME-XML metadata into the TIFF comment. The complicated part is doing the conversion from proprietary third-party metadata into OME-XML—a task that Bio-Formats greatly simplifies.

The following program converts the files given on the command line into OME-TIFF format. It requires the Bio-Formats and *OME-XML Java* libraries.

ConvertToOmeTiff.java

The code functions by creating an ImageReader for reading the input files' image planes sequentially, and an OMETiffWriter for writing the planes to OME-TIFF files on disk. The OME-XML is generated by attaching an OMEXMLMetadata object to the reader, such that when each file is initialized, the object is automatically populated with the converted metadata. The OMEXMLMetadata object is then fed to the OMETiffWriter, which extracts the appropriate OME-XML string and embeds it into the OME-TIFF file properly.

While our ultimate goal is for the Bio-Formats metadata conversion facility to be a reference implementation for conversion of third-party formats into OME-XML and OME-TIFF, please be aware that the current code is a work in progress. We would greatly value suggestions and assistance regarding the OME-XML conversion relating to any specific format. If there is any metadata missing or converted incorrectly, please let us know.

See also:

Exporting raw pixel data to OME-TIFF files and Converting files from FV1000 OIB/OIF to OME-TIFF

1.1.4 OME-TIFF sample data

This section provides some sample data in OME-TIFF format. They include data produced from an acquisition system as well as artificial sample datasets, i.e. designed for developer testing that illustrate some possible data organizations, which should be useful if you are interested in implementing support for OME-TIFF within your software.

All the OME-TIFF sample data discussed below are available from our OME-TIFF sample images resource and licensed under Creative Commons Attribution 4.0 International License unless specified otherwise.

Biological datasets

5D datasets

The following OME-TIFF datasets consist of tubulin histone GFP coexpressing *C. elegans* embryos. Many thanks to Josh Bembenek for preparing and imaging this sample data.

The datasets were acquired on a multiphoton workstation (2.1 GHz Athlon XP 3200+ with 1GB of RAM) using Wisc-Scan. All image planes were collected at 512 × 512 resolution in 8-bit grayscale, with an integration value of 2.

The files available for download have been updated to the current schema version since their initial creation.

Dataset (zip bundle)	Image dimensions (XYZCT)	Number of files
Tubhiswt 2D (tubhiswt-2D.zip)	512 × 512 × 1 × 2 × 1	2
Tubhiswt 3D (tubhiswt-3D.zip)	512 × 512 × 1 × 2 × 20	2
Tubhiswt 4D (tubhiswt-4D.zip)	512 × 512 × 10 × 2 × 43	86

Plate

Two OME-TIFF datasets representative of the *High Content Screening* section of the OME Data Model, derived from a 384 wells plate of the BBBC017 image set available from the Broad Bioimage Benchmark Collection.

Dataset	Image dimensions (XYZCT)	Provenance	Copyright
Single file OME-TIFF	512 × 512 × 1 × 3 × 1	BBBC017	CC-BY-NC-SA 3.0
Multi-file OME-TIFF	512 × 512 × 1 × 3 × 1	BBBC017	CC-BY-NC-SA 3.0

See Ljosa V, Sokolnicki KL, Carpenter AE (2012). Annotated high-throughput microscopy image sets for validation. Nature Methods 9(7):637.

Note: An OME-TIFF file representative of the same plate had been previously generated and made available under NIRHTa-001.ome.tiff. Although the file is syntactically valid, the plate layout is incorrect due to a conversion issue. This file should be considered as deprecated and superseded by the two representative plate examples described above.

ROI

An OME-TIFF dataset representative of the *ROI* section of the OME Data Model, derived from the public MitoCheck project.

Dataset	Image dimensions (XYZCT)	Provenance	Copyright
00001_01.ome.tiff	1344 × 1024 × 1 × 1 × 93	IDR	Public Domain

See Neumann B et al. (2010). Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. Nature 464(7289):721.

Sub-resolutions

A set of OME-TIFF datasets representative of the *Sub-resolutions* section of the OME-TIFF specification were derived from a few large X Y images.

Dataset	Image dimensions (XYZCT)	Notes	Provenance	Copyright
Leica-1.ome.tiff	36832 × 38432 × 1 × 3 × 1	WSI (Whole Slide Image), 2 RGB images	OpenSlide	Public Domain
Leica-2.ome.tiff	39168 × 26048 × 1 × 3 × 1	WSI, 5 RGB images	OpenSlide	Public Domain
LuCa-7color_Scan1.ome.tiff	24960 × 34560 × 1 × 5 × 1	WSI, multi-channel, fluorescence	PerkinElme	CC-BY 4.0
BGal_000438_frames.om	7676 × 7420 × 38 × 1 × 1	EM, Floating-point	EMPIAR	Public Domain
retina_large.ome.tiff	2048 × 1567 × 64 × 2 × 1	Multi-Z stack	Bitplane	CC-BY 4.0

Artificial datasets

5D datasets

All datasets in the following table are single OME-TIFF files generated using Bio-Formats `loci.formats.tools.MakeTestOmeTiff`. Each plane is labeled according to its dimensional position for easy testing.

Name	Image dimensions (XYZCT)	Available extensions
Single channel	439 × 167 × 1 × 1 × 1	ome.tif, ome.tiff, ome.tf8, ome.btf, ome.tf2
Multi channel	439 × 167 × 1 × 3 × 1	ome.tif, ome.tiff, ome.tf8, ome.btf, ome.tf2
Z series	439 × 167 × 5 × 1 × 1	ome.tif, ome.tiff, ome.tf8, ome.btf, ome.tf2
Time series	439 × 167 × 1 × 1 × 7	ome.tif, ome.tiff, ome.tf8, ome.btf, ome.tf2
Multi channel Z series	439 × 167 × 5 × 3 × 1	ome.tif, ome.tiff, ome.tf8, ome.btf, ome.tf2
Multi channel time series	439 × 167 × 1 × 3 × 7	ome.tif, ome.tiff, ome.tf8, ome.btf, ome.tf2
4D series	439 × 167 × 5 × 1 × 7	ome.tif, ome.tiff, ome.tf8, ome.btf, ome.tf2
Multi channel 4D series	439 × 167 × 5 × 3 × 7	ome.tif, ome.tiff, ome.tf8, ome.btf, ome.tf2

Modulo datasets

Sample files implementing the *6D, 7D and 8D storage* are available from the https://downloads.openmicroscopy.org/images/OME-TIFF/2016-06/modulo folder of the image downloads resource.

Name	Image dimensions (XYZCT)	Modulo description
SPIM-ModuloAlongZ.ome.tiff	160 × 220 × 8 × 2 × 12	4 tiles interleaved as ModuloAlongT each recorded at 4 angles interleaved as ModuloAlongZ
LAMBDA-ModuloAlongZ-ModuloAlongT.ome.tiff	200 × 200 × 5 × 1 × 10	excitation of 5 wavelength [, big-lambda] interleaved as ModuloAlongZ, each recorded at 10 emission wavelength ranges [, lambda] interleaved as ModuloAlongT
FLIM-ModuloAlongT-TSCPC.ome.tiff	180 × 220 × 1 × 2 × 16	2 channels and 8 histogram bins each recorded at 2 'real-time' points T, with additional relative-time points (time relative to the excitation pulse) interleaved as ModuloAlongT
FLIM-ModuloAlongC.ome.tiff	180 × 150 × 1 × 16 × 1	2 real channels and 8 histogram bins each recorded at 2 timepoints, with additional relative-time points interleaved between channels as ModuloAlongC

Multi-file OME-TIFF filesets

This section lists various examples of OME-TIFF datasets distributed across multiple TIFF files.

The first two datasets contain a set of 18 × 24 pixel images with black and white text on each plane giving its time, z-depth and channel. Each of the five focal planes is saved as a separate OME-TIFF named `multifile-Zxx.ome.tiff` where *xx* is the index of the focal plane.

The third dataset contains a plate with 4 wells at position A2, B1, B3 and C2. The first three wells contain one field of view and the fourth well contains 2 fields of view. Each well sample is saved as a separate OME-TIFF.

Dataset	Image dimensions (XYZCT)	Full metadata file*	Partial metadata files†
Master OME-TIFF fileset	18 × 24 × 5 × 1 × 1	`multifile-Z1.ome.tiff`	`multifile-Z[2-5].ome.tiff`
Companion OME-TIFF fileset	18 × 24 × 5 × 1 × 1	`multifile.companion.ome`	`multifile-Z[1-5].ome.tiff`
Companion OME-TIFF plate	96 × 96 × 1 × 1 × 1	`hcs.companion.ome`	`well-*.ome.tiff`

*

The full OME-XML metadata describing the whole fileset is either embedded into an OME-TIFF or stored in a companion OME-XML file

†

Partial OME-XML metadata blocks are embedded into the OME-TIFF files and refer to the file containing the full OME-XML metadata as described in the *specification*.

The OME-TIFF format was created to maximize the respective strengths of OME-XML and TIFF. It takes advantage of the rich metadata defined in OME-XML while retaining the pixels in multi-page TIFF format for compatibility with many more applications.

1.1.5 Characteristics

An OME-TIFF dataset has the following characteristics:

1. Image planes are stored within one multi-page TIFF file, or across multiple TIFF files. Any image organization is feasible.
2. A complete OME-XML metadata block describing the dataset is embedded in each TIFF file's header. Thus, even if some of the TIFF files in a dataset are misplaced, the metadata remains intact.
3. The OME-XML metadata block may contain anything allowed in a standard OME-XML file.
4. OME-TIFF uses the standard TIFF mechanism for storing one or more image planes in each of the constituent files, instead of encoding pixels as base64 chunks within the XML. Since TIFF is an image format, it makes sense to only use OME-TIFF as opposed to OME-XML, when there is at least one image plane.

1.1.6 Support

OME-TIFF is supported by:

- BIOVIA
- Bitplane AG
- Carl Zeiss Microscopy GmbH
- Cytiva (formerly GE Healthcare, Applied Precision)
- Definiens
- DRVision Technologies LLC
- Image-Pro by Media Cybernetics, Inc.
- iMagic
- Intelligent Imaging Innovations, Inc.
- Leica Inc.
- Mayachitra Inc.
- Micro-Manager
- Molecular Devices Inc.
- PerkinElmer
- Scientifica
- Scientific Volume Imaging B.V.
- Strand Life Sciences
- TILL Photonics GmbH, now FEI Munich

1.1.7 Public image repositories allowing image downloads as OME-TIFF

- ASCB CELL Image Library
- Harvard Medical School LINCS Project
- Stowers Institute Original Data Repository

CHAPTER

TWO

OME-XML

2.1 The OME-XML format

2.1.1 Extracting, processing and validating OME-XML

Extracting the OME-XML from an OME-TIFF file

If you install the Bio-Formats command line tools, you can produce a nicely formatted OME-XML string from an OME-TIFF file with:

```
$ tiffcomment file.ome.tif | xmlindent
```

Alternatively, if you have ImageMagick installed, one easy way to extract the OME-XML embedded in the TIFF headers is to use it from the command line:

```
$ identify -verbose
```

If you are working in C/C++, we recommend the open source LibTIFF library or the OME Files C++ implementation.

If you are looking for a solution in Java, there are several options. Bio-Formats can read OME-TIFF files, as well as convert from many third-party formats into OME-TIFF format—see the *example source code page* for specific examples. Alternatively, the open source ImageJ application reads multi-page TIFF files, storing the TIFF comment into the associated FileInfo object's "description" field.

Processing an OME-XML block

If the XML was stored without line breaks, it can still be difficult to read after being extracted. There are several solutions to this problem, such as using an XML viewer or editor (web browsers work well), or processing the XML with a SAX or DOM library.

On most Linux distributions, you can install the libxml package and use the `xmllint` program:

```
$ xmllint --format file.xml
```

Here is a Perl script that uses XML::LibXML to "pretty print" an XML document with appropriate whitespace:

```
formatxml.pl

use XML::LibXML;
$file = $ARGV[0];
$parser = XML::LibXML->new(); die "Cannot create XML parser" unless defined $parser;
```

(continues on next page)

(continued from previous page)

```
$parser->validation(0);
if (defined $file) { $doc = $parser->parse_file($file); }
else { $doc = $parser->parse_fh(STDIN); } print $doc->toString(1);
```

Unfortunately, both `xmllint` and the above Perl script can be somewhat fragile; if there are any errors or abnormalities in the XML, they generally fail to produce any indentation. Thus, we have also written some Java code to do the same thing; just download the Bio-Formats command line tools and run:

```
$ xmlindent file.xml
```

Another option is to feed the XML into our *OME-XML Java library*, which provides methods for querying and manipulating the OME-XML (using DOM and SAX). This library is what Bio-Formats uses to work with OME-XML.

Validating OME-XML

We have created a command line tool in Java for validating OME-XML, and included it as part of the Bio-Formats bftools.zip download. Please refer to the Bio-Formats command line tools documentation for more details but, in brief, you download and unzip the tools to produce a collection of command line scripts for Unix/Mac and batch files for Windows. The two commands we will use are:

xmlvalid
A command line XML validation tool

tiffcomment
Extracts the OME-XML block in an OME-TIFF file from the comment in the TIFF's first IFD entry.

All scripts require `bioformats_package.jar` to be downloaded into the same directory as the command line tools. Then to validate an OME-XML file `sample.ome` use:

```
$ xmlvalid sample.ome
```

This validates the XML directly.

Then to validate an OME-TIFF file `sample.ome.tif` use:

```
$ tiffcomment sample.ome.tif | xmlvalid
```

This extracts the OME-XML from the TIFF then passes it to the validator. Typical successful output is:

```
$ ./xmlvalid sample.ome
Parsing schema path
http://www.openmicroscopy.org/Schemas/OME/2010-06/ome.xsd
Validating sample.ome
No validation errors found.
$
```

If any errors are found they are reported. When correcting errors, it is usually best to work from the top of the file as errors higher up can cause extra errors further down. In this example the output shows 3 errors but there are only 2 mistakes in the file.

```
$ ./xmlvalid broken.ome
Parsing schema path
http://www.openmicroscopy.org/Schemas/OME/2010-06/ome.xsd
Validating broken.ome
```

(continues on next page)

(continued from previous page)

```
cvc-complex-type.4: Attribute 'SizeY' must appear on element 'Pixels'.
cvc-enumeration-valid: Value 'Non Zero' is not facet-valid with respect
   to enumeration '[EvenOdd, NonZero]'. It must be a value from the enumeration.
cvc-attribute.3: The value 'Non Zero' of attribute 'FillRule' on element
   'ROI:Shape' is not valid with respect to its type, 'null'.
Error validating document: 3 errors found
$
```

Alternatively you can chose a freely available online validator to validate your OME-XML blocks e.g. one from www.utilities-online.info .

Another option is to use a commercial XML application such as Turbo XML to work with and validate your OME-XML documents.

2.1.2 OME-XML Java library

The OME-XML Java library is a collection of Java packages for manipulating OME-XML metadata structures. The OME-XML Java library's metadata processing facilities form the backbone of the Bio-Formats library's support for OME-XML conversion.

Download

The `ome-xml` artifacts are hosted on the Maven Central Repository under the group `org.openmicroscopy` and distributed under the 2-Clause BSD license.

Installation

To use, add **ome-xml.jar** to your classpath or build path.

Usage

Refer to the online API documentation, specifically the ome.xml.* packages. For an example of usage, see the Screen Plate Well unit test.

The OMENode is the root ("OME") node of the OME-XML. Each XML element has its own node type (e.g. "Image" has ImageNode) with its own accessor and mutator methods, to make navigation of the OME-XML structure easier than it would be with a raw DOM object. However, there are some limitations to what can be done with the API. If your application needs access to a node's backing DOM element to work with it directly, you can call `getDOMElement()` on a node.

Source code

The OME-XML Java library is an open source project—the source code is freely accessible via the https://github.com/ome/ome-model Git repository.

OME-XML is a file format for storing microscopy information (both pixels and metadata) using the OME Data Model.

Note: OME-XML as a file format is superseded by OME-TIFF, which is the preferred container format for image data making use of the OME Data Model.

The purpose of OME-XML is to provide a rich, extensible way to save information concerning microscopy experiments and the images acquired therein, including:

- dimensional parameters defining the scope of the image pixels (e.g. resolution, number of focal planes, number of time points, number of channels)
- the hardware configuration used to acquire the image planes (e.g. microscope, detectors, lenses, filters)
- the settings used with said hardware (e.g. physical size of the image planes in microns, laser gain and offset, channel configuration)
- the person performing the experiment
- some details regarding the experiment itself, such as a description, the type of experiment (e.g. FRET, time lapse, fluorescence lifetime) and events occurring during acquisition (e.g. laser ablation, stage motion)
- additional custom information you may wish to provide about your experiment in a structured form (known as *Structured Annotations*)

2.1.3 Features and applications

The OME-XML file serves as a convenient file format for data migration from one site or user to another. The OME-XML file captures all image acquisition and experimental metadata, along with the binary image data, and packages it into an easily readable package. The paper describing the design and implementation of the OME-XML file appeared in Genome Biology.

Note: OME-XML files can be read by potentially any software package - you do not need OME image management software to use OME-XML.

Some specific features of the OME-XML file format:

- OME-XML files may contain one or more sets of 5-D pixels, for example raw data from a microscope, the deconvolved data, and a volume rendered view.
- OME-XML files contain all the metadata associated with an image, including the experimental (e.g. cells, genes) and acquisition (e.g. microscope light sources, filters, detectors) metadata.
- OME-XML Image pixels may be stored compressed directly in XML with base64 encoding. Compression of the pixels and the metadata is supported through widely-available patent-free compression schemes (gzip and bzip2). OME-XML Images are addressable by plane.
- OME-XML files have a built-in mechanism for supporting arbitrary user-defined data that can be used globally or attached to Images, Features (objects inside Images), and Datasets. Mechanisms for OME-compliant systems to populate databases with these user-defined fields is part of the specification. See the *Structured Annotations* XML Schema section.

The OME-XML Schema .xsd files and technical documentation are available on the Schema pages.

CHAPTER

THREE

OMERO PYRAMID

3.1 The OMERO pyramid format

The OMERO pyramid format is a way of storing very large images for easier visualization. Currently only v1.0.0 is defined.

See also:

Working with whole slide images

3.1.1 v1.0.0

The OMERO pyramid format is a TIFF file containing JPEG-2000 compressed image tiles. All resolutions for a tile are encoded in the same JPEG-2000 stream, using the "decompression levels" feature of JPEG-2000. As a result, only data types supported by the JPEG-2000 standard (`uint8` and `uint16`) are supported. Images with pixel type `uint32`, `float` (32-bit floating point), or `double` (64-bit floating point) cannot be converted to an OMERO pyramid. Pyramid files larger than 4 gigabytes are supported, as are pyramids containing multiple channels, Z sections, and/or timepoints.

Each pyramid contains multiple resolutions for each image plane, with each resolution stored in descending order from largest to smallest XY size. Each resolution is half the width and height of the previous resolution. OMERO by default writes 5 resolutions, but this is an implementation detail and not a limitation of the file format.

One IFD is required to be stored for each image plane, but every resolution for a given plane is encapsulated in that plane's single IFD. Additional IFDs for each resolution are not expected; any IFDs that do not represent a JPEG-2000 stream with multiple decompression levels will be ignored.

CHAPTER

FOUR

FILE SPECIFICATIONS

4.1 Compliant file specification

4.1.1 Compliant HCS specification

OME compliant HCS file specification. This was developed in conjunction with the June 2010 release of the OME-XML Model and has been updated to the June 2016 release.

A compliant specification for HCS OME files has been defined. This is not the minimum required for the display of an image, that is the *Minimum Specification*. This is the information a file should contain to be useful while conforming to the spirit of the OME projects aims.

This sample structure is based on Sample One in the OME Compliant File Specification (see *Compliant file specification*):

```
<?xml version="1.0" encoding="UTF-8"?>
<OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06
                         http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd">

  <Plate
     ID="Plate:1"
     Name="Control Plate"
     ColumnNamingConvention="letter"
     RowNamingConvention="number"
     Columns="12"
     Rows="8"
     >
    <Description></Description>

    <!-- repeat Well for # of wells in the plate that contain images -->
    <Well ID="Well:1" Column="0" Row="0">
      <!-- repeat WellSample for # of images taken in the well -->
      <WellSample ID="WellSample:1" Index="0">
        <!--
             if there is an image associated with this SPW:WellSample
             it is linked using an ImageRef
          -->
        <ImageRef ID="Image:0"/>
      </WellSample>
```

(continues on next page)

(continued from previous page)

```
    </Well>
  </Plate>
  <!-- plus one more Plate for each Plate in this set -->

  <!-- Screen is not required -->

  <!-- The Image element follows the structure for the OME Compliant File Specification -
↪->
  <Image ID="Image:0" Name="Series 1">
    <AcquisitionDate>2008-02-06T13:43:19</AcquisitionDate>
    <Description>An example OME compliant file, based on Olympus.oib</Description>
    <Pixels DimensionOrder="XYCZT" ID="Pixels:0"
                PhysicalSizeX="0.207" PhysicalSizeY="0.207"
                SizeC="3" SizeT="16" SizeX="1024" SizeY="1024" SizeZ="1"
                TimeIncrement="120.1302" Type="uint16">
      <Channel EmissionWavelength="523" ExcitationWavelength="488" ID="Channel:0:0"
                  IlluminationType="Epifluorescence" Name="CH1" SamplesPerPixel="1"
                  PinholeSize="103.5" AcquisitionMode="LaserScanningConfocalMicroscopy"/
↪>
      <Channel EmissionWavelength="578" ExcitationWavelength="561" ID="Channel:0:1"
                  IlluminationType="Epifluorescence" Name="CH3" SamplesPerPixel="1"
                  PinholeSize="127.24" AcquisitionMode="LaserScanningConfocalMicroscopy
↪"/>
      <Channel ExcitationWavelength="488" ID="Channel:0:2"
                  IlluminationType="Transmitted"
                  ContrastMethod="DIC" Name="TD1" SamplesPerPixel="1"
                  AcquisitionMode="LaserScanningConfocalMicroscopy"/>
      <BinData BigEndian="false" Length="0"/>
    </Pixels>
  </Image>
</OME>
```

Alternative valid forms:

- would have a `TiffData` block instead of the `BinData` block (this would be used in the header of an OME-TIFF file)
- would have a `MetadataOnly` block instead of the `BinData` block (this would be used as a companion to one or more `BinaryOnly` OME-TIFF files)

Note: A units system was added in January 2015. This sample assumes the default unit for each value is used.

4.1.2 Minimum specification

This was developed in conjunction with the September 2009 release of the OME-XML Model and has been updated to the June 2016 release.

A minimum specification OME file has been defined. This is best viewed as being the minimum required for the display of an image. A sample of the structure is:

```
<?xml version="1.0" encoding="UTF-8"?>
<OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06
                         http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd">
  <Image ID="Image:1" Name="Name92">
    <Pixels
       ID="Pixels:1"
       DimensionOrder="XYZCT"
       Type="int8"
       SizeX="2"
       SizeY="2"
       SizeZ="2"
       SizeC="2"
       SizeT="2">
      <BinData
         BigEndian="false"
         Compression="none"
         Length="12"
         >ZGVmYXVsdA==</BinData>
    </Pixels>
  </Image>
</OME>
```

Alternative valid forms:

- would have a <TiffData/> block instead of the BinData block (this would be used in the header of an OME-TIFF file)
- would have a <MetadataOnly/> block instead of the BinData block (this would be used as a companion to one or more BinaryOnly OME-TIFF files)

Note: A units system was added in January 2015. This sample assumes the default unit for each value is used.

This was developed in conjunction with the April 2010 release of the OME-XML Model and the samples have been updated to the June 2016 release.

A "compliant" specification OME file has been defined. This is not the minimum required for the display of an image (for this, see *Minimum Specification*). This is the information a file should contain to authoritatively describe an imaging experiment, so that another person could, with the same sample and microscope, reproduce the data recorded in the file. Therefore, an 'OME Compliant' file should be as complete as possible, but only contain metadata relevant to a specific imaging experiment i.e. all of the fields listed below that are relevant to the experiment should be completed. As an example, PockelCellSetting would not be used in most wide-field microscopy experiments, but would be mandatory for many multi-photon imaging experiments.

4.1.3 Sample structure

Two samples of this structure are shown below. They were generated by opening a propriety file (Olympus .oib and DeltaVision .dv, respectively) in ImageJ, using the Bio-Formats plugin with the *Display OME-XML Metadata* option checked. A few additional attributes were then manually added to the files to make them compliant.

Sample structure one:

```
<?xml version="1.0"?>
<OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:str="http://exslt.org/strings"
     xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06                ␣
↪                http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd">

  <Image ID="Image:0" Name="Series 1">
    <AcquisitionDate>2008-02-06T13:43:19</AcquisitionDate>
    <Description>An example OME compliant file, based on Olympus.oib</Description>
    <Pixels DimensionOrder="XYCZT" ID="Pixels:0" PhysicalSizeX="0.207" PhysicalSizeY="0.
↪207" SizeC="3" SizeT="16" SizeX="1024" SizeY="1024" SizeZ="1" TimeIncrement="120.1302"␣
↪Type="uint16">
      <Channel EmissionWavelength="523" ExcitationWavelength="488" ID="Channel:0:0"␣
↪IlluminationType="Epifluorescence" Name="CH1" SamplesPerPixel="1" PinholeSize="103.5"␣
↪AcquisitionMode="LaserScanningConfocalMicroscopy"/>
      <Channel EmissionWavelength="578" ExcitationWavelength="561" ID="Channel:0:1"␣
↪IlluminationType="Epifluorescence" Name="CH3" SamplesPerPixel="1" PinholeSize="127.24"␣
↪AcquisitionMode="LaserScanningConfocalMicroscopy"/>
      <Channel ExcitationWavelength="488" ID="Channel:0:2" IlluminationType="Transmitted
↪" ContrastMethod="DIC" Name="TD1" SamplesPerPixel="1" AcquisitionMode=
↪"LaserScanningConfocalMicroscopy"/>
      <BinData BigEndian="false" Length="0"/>
    </Pixels>
  </Image>
</OME>
```

Sample structure two:

```
<?xml version="1.0"?>
<OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:str="http://exslt.org/strings"
     xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06                ␣
↪                http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd">

  <Instrument ID="Instrument:0">
    <Detector ID="Detector:0:0" Model="COOLSNAP_HQ / ICX285" Type="CCD"/>
    <Objective ID="Objective:10002" Immersion="Oil" LensNA="1.4" Manufacturer="Olympus"␣
↪NominalMagnification="100"/>
  </Instrument>
  <Image ID="Image:1" Name="example_R3D_D3D.dv">
    <AcquisitionDate>2005-01-28T13:50:08</AcquisitionDate>
    <Description>An example OME compliant file,
            based on a wide-field microscope image</Description>
    <ObjectiveSettings ID="Objective:10002" Medium="Oil" RefractiveIndex="1.52"␣
```

(continues on next page)

(continued from previous page)

```
↪CorrectionCollar="7"/>
    <Pixels DimensionOrder="XYCZT" ID="Pixels:1" PhysicalSizeX="0.06631" PhysicalSizeY=
↪"0.06631" PhysicalSizeZ="0.2" SizeC="3" SizeT="1" SizeX="480" SizeY="480" SizeZ="5"␣
↪Type="int16">
      <Channel EmissionWavelength="457" ExcitationWavelength="360" ID="Channel:1:0"␣
↪NDFilter="0.5" Name="DAPI" SamplesPerPixel="1" Fluor="DAPI" IlluminationType=
↪"Epifluorescence" ContrastMethod="Fluorescence" AcquisitionMode="WideField" Color=
↪"65535">
        <DetectorSettings Binning="1x1" Gain="0.5" ID="Detector:0:0" ReadOutRate="10.0"/>
      </Channel>
      <Channel EmissionWavelength="528" ExcitationWavelength="490" ID="Channel:1:1"␣
↪NDFilter="0.0" Name="FITC" SamplesPerPixel="1" Fluor="GFP" IlluminationType=
↪"Epifluorescence" ContrastMethod="Fluorescence" AcquisitionMode="WideField" Color=
↪"16711935">
        <DetectorSettings Binning="1x1" Gain="0.5" ID="Detector:0:0" ReadOutRate="10.0"/>
      </Channel>
      <Channel EmissionWavelength="617" ExcitationWavelength="555" ID="Channel:1:2"␣
↪NDFilter="0.0" Name="RD-TR-PE" SamplesPerPixel="1" Fluor="TRITC" IlluminationType=
↪"Epifluorescence" ContrastMethod="Fluorescence" AcquisitionMode="WideField" Color="-
↪16776961">
        <DetectorSettings Binning="1x1" Gain="0.5" ID="Detector:0:0" ReadOutRate="10.0"/>
      </Channel>
      <BinData BigEndian="false" Length="0"/>
      <Plane DeltaT="0.0" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-21.496" TheC="0" TheT="0" TheZ="0"/>
      <Plane DeltaT="0.294" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-21.696" TheC="0" TheT="0" TheZ="1"/>
      <Plane DeltaT="0.587" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-21.896" TheC="0" TheT="0" TheZ="2"/>
      <Plane DeltaT="0.881" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-22.096" TheC="0" TheT="0" TheZ="3"/>
      <Plane DeltaT="1.174" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-22.296" TheC="0" TheT="0" TheZ="4"/>
      <Plane DeltaT="9.625" ExposureTime="0.3" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-21.496" TheC="1" TheT="0" TheZ="0"/>
      <Plane DeltaT="10.12" ExposureTime="0.3" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-21.696" TheC="1" TheT="0" TheZ="1"/>
      <Plane DeltaT="10.613" ExposureTime="0.3" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-21.896" TheC="1" TheT="0" TheZ="2"/>
      <Plane DeltaT="11.106" ExposureTime="0.3" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-22.096" TheC="1" TheT="0" TheZ="3"/>
      <Plane DeltaT="11.599" ExposureTime="0.3" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-22.296" TheC="1" TheT="0" TheZ="4"/>
      <Plane DeltaT="25.447" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-21.496" TheC="2" TheT="0" TheZ="0"/>
      <Plane DeltaT="25.739" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-21.696" TheC="2" TheT="0" TheZ="1"/>
      <Plane DeltaT="26.033" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-21.896" TheC="2" TheT="0" TheZ="2"/>
      <Plane DeltaT="26.326" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
↪PositionZ="-22.096" TheC="2" TheT="0" TheZ="3"/>
      <Plane DeltaT="26.619" ExposureTime="0.1" PositionX="3316.37" PositionY="-646.46"␣
```

(continues on next page)

(continued from previous page)

```
↪PositionZ="-22.296" TheC="2" TheT="0" TheZ="4"/>
    </Pixels>
  </Image>
</OME>
```

Note: The data (BinData element content) in these sample was removed for reasons of length.

Alternative valid forms:

- would have a `TiffData` block instead of the `BinData` block (this would be used in the header of an OME-TIFF file)
- would have a `MetadataOnly` block instead of the `BinData` block (this would be used as a companion to one or more `BinaryOnly` OME-TIFF files)

Note: A units system was added in January 2015. This sample assumes the default unit for each value is used.

4.1.4 Definitions of values stored

The figure *Definitions of values stored* is ©2010 Linkert et al. Figure originally published as Figure 2, Linkert et al (2010), J. Cell Biol. 189(5):777-782 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2878938/)

Note: AcquiredDate was renamed to AcquisitionDate in June 2012

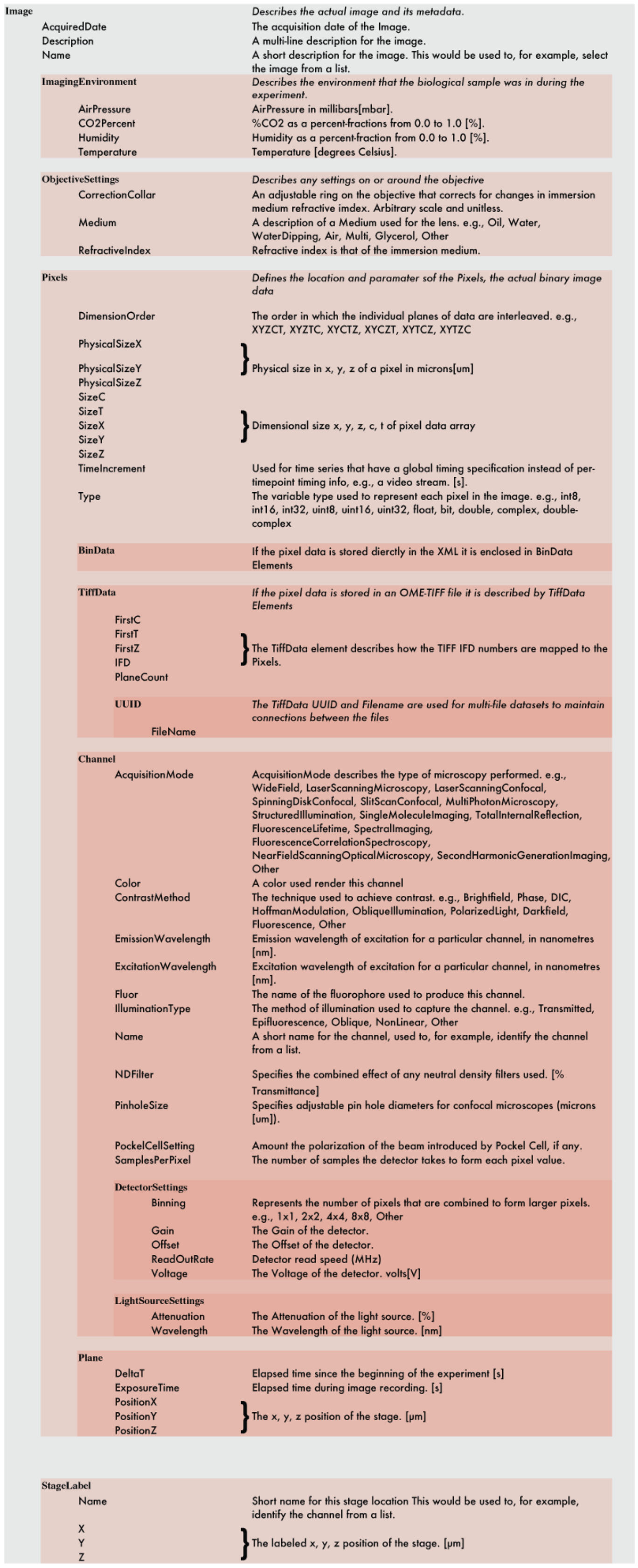


Image	*Describes the actual image and its metadata.*
AcquiredDate	The acquisition date of the Image.
Description	A multi-line description for the image.
Name	A short description for the image. This would be used to, for example, select the image from a list.
ImagingEnvironment	*Describes the environment that the biological sample was in during the experiment.*
AirPressure	AirPressure in millibars[mbar].
CO2Percent	%CO2 as a percent-fractions from 0.0 to 1.0 [%].
Humidity	Humidity as a percent-fraction from 0.0 to 1.0 [%].
Temperature	Temperature [degrees Celsius].
ObjectiveSettings	*Describes any settings on or around the objective*
CorrectionCollar	An adjustable ring on the objective that corrects for changes in immersion medium refractive imdex. Arbitrary scale and unitless.
Medium	A description of a Medium used for the lens. e.g., Oil, Water, WaterDipping, Air, Multi, Glycerol, Other
RefractiveIndex	Refractive index is that of the immersion medium.
Pixels	*Defines the location and paramater sof the Pixels, the actual binary image data*
DimensionOrder	The order in which the individual planes of data are interleaved. e.g., XYZCT, XYZTC, XYCTZ, XYCZT, XYTCZ, XYTZC
PhysicalSizeX PhysicalSizeY PhysicalSizeZ	Physical size in x, y, z of a pixel in microns[um]
SizeC SizeT SizeX SizeY SizeZ	Dimensional size x, y, z, c, t of pixel data array
TimeIncrement	Used for time series that have a global timing specification instead of per-timepoint timing info, e.g., a video stream. [s].
Type	The variable type used to represent each pixel in the image. e.g., int8, int16, int32, uint8, uint16, uint32, float, bit, double, complex, double-complex
BinData	If the pixel data is stored dierctly in the XML it is enclosed in BinData Elements
TiffData	*If the pixel data is stored in an OME-TIFF file it is described by TiffData Elements*
FirstC FirstT FirstZ IFD PlaneCount	The TiffData element describes how the TIFF IFD numbers are mapped to the Pixels.
UUID	*The TiffData UUID and Filename are used for multi-file datasets to maintain connections between the files*
FileName	
Channel	
AcquisitionMode	AcquisitionMode describes the type of microscopy performed. e.g., WideField, LaserScanningMicroscopy, LaserScanningConfocal, SpinningDiskConfocal, SlitScanConfocal, MultiPhotonMicroscopy, StructuredIllumination, SingleMoleculeImaging, TotalInternalReflection, FluorescenceLifetime, SpectralImaging, FluorescenceCorrelationSpectroscopy, NearFieldScanningOpticalMicroscopy, SecondHarmonicGenerationImaging, Other
Color	A color used render this channel
ContrastMethod	The technique used to achieve contrast. e.g., Brightfield, Phase, DIC, HoffmanModulation, ObliqueIllumination, PolarizedLight, Darkfield, Fluorescence, Other
EmissionWavelength	Emission wavelength of excitation for a particular channel, in nanometres [nm].
ExcitationWavelength	Excitation wavelength of excitation for a particular channel, in nanometres [nm].
Fluor	The name of the fluorophore used to produce this channel.
IlluminationType	The method of illumination used to capture the channel. e.g., Transmitted, Epifluorescence, Oblique, NonLinear, Other
Name	A short name for the channel, used to, for example, identify the channel from a list.
NDFilter	Specifies the combined effect of any neutral density filters used. [% Transmittance]
PinholeSize	Specifies adjustable pin hole diameters for confocal microscopes (microns [um]).
PockelCellSetting	Amount the polarization of the beam introduced by Pockel Cell, if any.
SamplesPerPixel	The number of samples the detector takes to form each pixel value.
DetectorSettings	
Binning	Represents the number of pixels that are combined to form larger pixels. e.g., 1x1, 2x2, 4x4, 8x8, Other
Gain	The Gain of the detector.
Offset	The Offset of the detector.
ReadOutRate	Detector read speed (MHz)
Voltage	The Voltage of the detector. volts[V]
LightSourceSettings	
Attenuation	The Attenuation of the light source. [%]
Wavelength	The Wavelength of the light source. [nm]
Plane	
DeltaT	Elapsed time since the beginning of the experiment [s]
ExposureTime	Elapsed time during image recording. [s]
PositionX PositionY PositionZ	The x, y, z position of the stage. [μm]
StageLabel	
Name	Short name for this stage location This would be used to, for example, identify the channel from a list.
X Y Z	The labeled x, y, z position of the stage. [μm]

Fig. 1: Definitions of values stored
©2010 Linkert et al. Figure originally published as Figure 2, Linkert et al (2010), J. Cell Biol. 189(5):777-782

CHAPTER

FIVE

DEVELOPER TOOLS AND GUIDELINES

5.1 Developer introduction

5.1.1 Using OME-XML schema elements

This sections explains how to use OME schema elements in your own XML files.

Note: The correct attribution for work derived from our schema files under the Creative Commons licence is: **"This work is derived in part from the OME specification. Copyright (C) 2002-2017 Open Microscopy Environment"**

We encourage people to make use of the OME Data Model as a whole by using the OME-XML and OME-TIFF formats directly. If you need to store additional user-specific data, the correct approach is to embed it within the OME Data Model using Structured Annotations. These provide a powerful and flexible way of storing your own custom data alongside the OME model metadata, while still maintaining full compatibility with OME-XML and OME-TIFF files, ensuring your file can be read by any application supporting these formats.

If this approach will not work for you, we would encourage you to contact us via the forums and mailing lists as we may be able to help you structure your additions so compatibility can be maintained. Fully compatible and exchangeable files is what this project all is about, so we try to support this as much as we can.

If there is still no possible approach for your data that can maintain compatibility with OME-XML or OME-TIFF, then what follows is the best practice for this situation.

Things to avoid

1. Adding arbitrary items to the OME Model and still calling it OME-XML or OME-TIFF

   We cannot stress enough that this is a **very bad** approach! If you simply insert your own custom nodes at arbitrary points within the XML document, you will produce a broken/invalid OME-XML or OME-TIFF file. As the outer wrapping looks like an OME file, people will expect it to work like an OME file. Applications are likely to fail to import the file as they will produce an error as they encounter your additional XML elements. This will frustrate end users and is likely to produce reports of broken applications. These tend to ultimately propagate back to our development team and we have to investigate, which eats into the time we have available to work on new features and formats.

2. Adding or removing arbitrary items to/from the OME Model by copying our structure and calling it something else

   This is also **not** a good approach. If you copy the OME Model and then make changes (whether by deletion or addition) you are also producing a broken/invalid file, even if you call it something else. We have defined parts of the model that can be omitted (marked as optional in the schema), and defined points where additions can be made (inside the Structured Annotations). It is important to us that we have some control over additions and omissions as it allows us to produce a format that has the widest compatibility. Even though the specifications we produce are now released under a less restrictive license, we do not encourage this approach. We want anyone who encounters an OME node in an XML document to be able to trust its structure, as well as validate and parse it.

3. A 'pick and mix' by copying our structure

   While we have changed the licence the schema files for the OME Model are released under to allow this approach, we would not encourage this. You should **not** need to copy pieces of our definitions and place them directly in your schema documents.

   Direct copying of our structure is not necessary as XML already has a method of including items from another schema, **see below**. From our point of view this is important as it allows us to control which parts of our model can be used as stand-alone objects in others' work.

Correct approaches

If you have to define your own schema that will make use of our OME model, there are two approaches:

1. Wrapping our entire OME Model in your custom model

   The entire OME Model, when used in an instance document, is completely contained inside the `<OME>` node. It is possible to include a complete `<OME>` node from the OME schema within your own custom XML node. While this does not maintain direct compatibility with the OME-XML and OME-TIFF file formats, it keeps all of the OME Model data together as a single block, that can be easily extracted from your custom XML nodes and passed to a standard OME-XML parser for it to interpret. It only takes a single line of XML Schema Description Language to include the whole model like this.

   We would always see this wrapping of the **entire OME model** as the best approach to take if you have to define your own custom model. It is also the best way of future-proofing yourself, as an `<OME>` node included like this will be easier to upgrade to newer schema releases using our standard transform.

2. A 'pick and mix' by referencing parts of our structure

   The least desirable valid approach is to individually include small independent parts of the model. Any of the items defined at the top level of the OME schema may be included individually within your custom model. If you take this approach, you must understand that including a node also includes those nodes below it, i.e. including LightSource also includes Laser, Arc etc. This approach does let you select individual parts of our OME model to include, but also lets us control which parts of the model are available for inclusion. Any reading/writing code for OME model pieces stored in this form will have to be custom written, as standard OME model parsers will not be able to process the pieces.

Note: With either of these approaches please acknowledge our work by including in the appropriate place in your software or project documentation: **"This work is derived in part from the OME specification. Copyright (C) 2002-2017 Open Microscopy Environment"**

Worked examples

Here are a few examples of how to define a link to the OME model from within your custom schema file and instance documents.

sample-third-party.xml

This is an instance document - the xml file that contains your data and is structured to conform to your custom schema specification:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomTag
    xmlns="http://www.example.org/SampleThirdPartySchema/2013-01"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.example.org/SampleThirdPartySchema/2013-01
        file:sample-third-party.xsd">

    <YourNodes>
        <With your="attributes"/>
    </YourNodes>

    <!-- Insert an OME node from the 2016-06 version of our schema -->
    <OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
        xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06
                            http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd"
        >
        <Image ID="Image:1">
            <AcquisitionDate>2010-02-23T12:51:30</AcquisitionDate>
            <Pixels ID="Pixels:1" DimensionOrder="XYZCT" Type="uint8" SizeX="1" SizeY="1
↪" SizeZ="1"
                SizeT="1" SizeC="1">
                <MetadataOnly/>
            </Pixels>
        </Image>
    </OME>
    <!-- Finish the OME node, and continue with your custom schema -->

    <MoreOfYourNodes></MoreOfYourNodes>
</CustomTag>
```

This file has `<YourNodes>` followed by the `<OME>` node, then `<MoreOfYourNodes>`. Apart from the xml namespace and schema location attributes on the `<OME>` node, the file is the same as though the OME model was part of your custom namespace.

sample-third-party.xsd

In order to define the easy-to-use structure described in the sample-third-party.xml file, you need to add three things (marked ****) to your schema specification document:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- **** Define the OME namespace for your schema on the <schema> node **** -->
<xs:schema
    xmlns:OME="http://www.openmicroscopy.org/Schemas/OME/2016-06"

    xmlns="http://www.example.org/SampleThirdPartySchema/2013-01"
    targetNamespace="http://www.example.org/SampleThirdPartySchema/2013-01"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    version="1"
    elementFormDefault="qualified">

    <!-- **** Include the OME namespace to make it accessible from your schema **** -->
    <xs:import namespace="http://www.openmicroscopy.org/Schemas/OME/2016-06"
    schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd"/>

    <xs:element name="CustomTag">
        <xs:annotation>
            <xs:documentation>
                Open Microscopy Environment
                OME Sample Third Party
                Copyright 2016 OME.
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="YourNodes">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="With">
                                <xs:complexType>
                                    <xs:attribute name="your" use="required" type=
↪"xs:string"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

                <!-- **** Reference to the OME element **** -->
                <xs:element ref="OME:OME" minOccurs="1" maxOccurs="1"/>

                <xs:element name="MoreOfYourNodes"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

sample-third-party-pieces.xml

If you want to import only a few pieces of the OME Model, this example illustrates how to include `<LightSource>` and `<Objective>`:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomTag
    xmlns="http://www.example.org/SampleThirdPartySchemaPieces/2013-01"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.example.org/SampleThirdPartySchemaPieces/2013-01
        file:sample-third-party-pieces.xsd">

    <YourNodes>
        <With your="attributes"/>
    </YourNodes>

    <!-- Insert a LightSource node from the 2016-06 version of our schema -->
    <Laser xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
           xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06
             http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd"
        ID="LightSource:1" Type="Dye" FrequencyMultiplication="2"
        LaserMedium="CoumarinC30" PockelCell="true" Pulse="Single"
        RepetitionRate="1.3" Tuneable="true" Wavelength="640">
      <Pump ID="LightSource:4"/>
    </Laser>
    <!-- Finish the LightSource node, and continue with your custom schema -->

    <MoreOfYourNodes>

        <!-- Insert an Objective node from the 2016-06 version of our schema -->
        <Objective xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
            xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06
                 http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd"
            ID="Objective:1" CalibratedMagnification="0.3" Correction="UV"
            Immersion="Air" Iris="true" LensNA="1.3" NominalMagnification="2"
            WorkingDistance="2.3" Manufacturer="OME-Sample" Model="Mk II"
            SerialNumber="sn-234567"/>
        <!-- Finish the Objective node, and continue with your custom schema -->

        <EvenMoreOfYourNodes></EvenMoreOfYourNodes>
    </MoreOfYourNodes>
</CustomTag>
```

sample-third-party-pieces.xsd

In order to define this sample-third-party-pieces.xml structure, you need to add four lines (marked ****) to your schema specification document. The first two lines are the same as the previous schema specification, then add one line for each of the two included nodes:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- **** Define the OME namespace for your schema on the <schema> node **** -->
<xs:schema
    xmlns:OME="http://www.openmicroscopy.org/Schemas/OME/2016-06"

    xmlns="http://www.example.org/SampleThirdPartySchemaPieces/2013-01"
    targetNamespace="http://www.example.org/SampleThirdPartySchemaPieces/2013-01"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    version="1"
    elementFormDefault="qualified">

    <!-- **** Include the OME namespace to make it accessible from your schema **** -->
    <xs:import namespace="http://www.openmicroscopy.org/Schemas/OME/2016-06"
    schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06/ome.xsd"/>

    <xs:element name="CustomTag">
        <xs:annotation>
            <xs:documentation>
                Open Microscopy Environment
                OME Sample Third Party
                Copyright 2016 OME.
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="YourNodes">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="With">
                                <xs:complexType>
                                    <xs:attribute name="your" use="required" type=
↪"xs:string"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

                <!-- **** Reference to the LightSource element **** -->
                <xs:element ref="OME:LightSource" minOccurs="1" maxOccurs="1"/>

                <xs:element name="MoreOfYourNodes">
                    <xs:complexType>
                        <xs:sequence>

                            <!-- **** Reference to the Objective element **** -->
```

(continues on next page)

(continued from previous page)

```
                            <xs:element ref="OME:Objective" minOccurs="1" maxOccurs="1"/>

                            <xs:element name="EvenMoreOfYourNodes"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Possible included objects

- `<xs:element ref="AnnotationRef">`
- `<xs:element ref="Arc">`
- `<xs:element ref="BinaryFile">`
- `<xs:element ref="BinData">`
- `<xs:element ref="BooleanAnnotation">`
- `<xs:element ref="Channel">`
- `<xs:element ref="ChannelRef">`
- `<xs:element ref="CommentAnnotation">`
- `<xs:element ref="Dataset">`
- `<xs:element ref="DatasetRef">`
- `<xs:element ref="Detector">`
- `<xs:element ref="DetectorSettings">`
- `<xs:element ref="Dichroic">`
- `<xs:element ref="DichroicRef">`
- `<xs:element ref="DoubleAnnotation">`
- `<xs:element ref="Ellipse">`
- `<xs:element ref="FilterRef">`
- `<xs:element ref="Experiment">`
- `<xs:element ref="Experimenter">`
- `<xs:element ref="ExperimenterGroup">`
- `<xs:element ref="ExperimenterGroupRef">`
- `<xs:element ref="ExperimenterRef">`
- `<xs:element ref="ExperimentRef">`
- `<xs:element ref="External">`
- `<xs:element ref="Filament">`
- `<xs:element ref="FileAnnotation">`
- `<xs:element ref="Filter">`
- `<xs:element ref="FilterSet">`
- `<xs:element ref="FilterSetRef">`
- `<xs:element ref="GenericExcitationSource">`
- `<xs:element ref="HashSHA1">`
- `<xs:element ref="Image">`
- `<xs:element ref="ImageRef">`
- `<xs:element ref="ImagingEnvironment">`
- `<xs:element ref="Instrument">`
- `<xs:element ref="InstrumentRef">`
- `<xs:element ref="Label">`
- `<xs:element ref="Laser">`
- `<xs:element ref="Leader">`
- `<xs:element ref="LightEmittingDiode">`
- `<xs:element ref="LightPath">`
- `<xs:element ref="LightSource">`
- `<xs:element ref="LightSourceSettings">`
- `<xs:element ref="Line">`
- `<xs:element ref="ListAnnotation">`
- `<xs:element ref="LongAnnotation">`
- `<xs:element ref="M">` *(part of the Map key/value structure)*
- `<xs:element ref="Map">`
- `<xs:element ref="MapAnnotation">`
- `<xs:element ref="Mask">`
- `<xs:element ref="MetadataOnly">`
- `<xs:element ref="MicrobeamManipulation">`
- `<xs:element ref="MicrobeamManipulationRef">`
- `<xs:element ref="Microscope">`
- `<xs:element ref="Objective">`
- `<xs:element ref="ObjectiveSettings">`
- `<xs:element ref="OME">`
- `<xs:element ref="Pixels">`
- `<xs:element ref="Plane">`
- `<xs:element ref="Plate">`
- `<xs:element ref="PlateAcquisition">`
- `<xs:element ref="PlateRef">`
- `<xs:element ref="Point">`
- `<xs:element ref="Polygon">`
- `<xs:element ref="Polyline">`
- `<xs:element ref="Project">`
- `<xs:element ref="ProjectRef">`
- `<xs:element ref="Pump">`
- `<xs:element ref="Reagent">`
- `<xs:element ref="ReagentRef">`
- `<xs:element ref="Rectangle">`
- `<xs:element ref="Rights">`

- `<xs:element ref="RightsHeld">`
- `<xs:element ref="RightsHolder">`
- `<xs:element ref="ROI">`
- `<xs:element ref="ROIRef">`
- `<xs:element ref="Screen">`
- `<xs:element ref="Shape">`
- `<xs:element ref="StageLabel">`
- `<xs:element ref="StructuredAnnotations">`
- `<xs:element ref="TagAnnotation">`
- `<xs:element ref="TermAnnotation">`
- `<xs:element ref="TiffData">`
- `<xs:element ref="TimestampAnnotation">`
- `<xs:element ref="TransmittanceRange">`
- `<xs:element ref="UUID">`
- `<xs:element ref="Well">`
- `<xs:element ref="WellSample">`
- `<xs:element ref="WellSampleRef">`
- `<xs:element ref="XMLAnnotation">`

5.1.2 File compression

This section provides an analysis of several file formats (including *OME-XML* and *OME-TIFF*) and compression techniques.

The figures regarding various storage formats were computed from the *5D datasets* before the current schema version. Thus, the byte counts between the downloadable ZIP files and the "zipped OME-TIFF" entry do not precisely match. The table below also lists the space requirements for each dataset with various formats and compression types.

Dataset	tubhiswt-2D.zip	tubhiswt-3D.zip	tubhiswt-4D.zip
Download Size	238,344 bytes	4,502,202 bytes	106,787,266 bytes
Size (raw pixels only)	524,288 bytes	10,485,760 bytes	225,443,840 bytes
Size (OME-XML)	314,346 bytes	5,964,603 bytes	142,498,355 bytes
Size (gzipped OME-XML)	236,708 bytes	4,511,329 bytes	107,788,464 bytes
Size (zipped OME-XML)	236,836 bytes	4,511,457 bytes	107,788,592 bytes
Size (7-zipped OME-XML)	239,052 bytes	4,551,263 bytes	108,708,700 bytes
Size (OME-TIFF)	531,384 bytes	10,499,384 bytes	225,874,680 bytes
Size (OME-TIFF with LZW)	273,190 bytes	4,998,148 bytes	118,517,497 bytes
Size (zipped OME-TIFF)	235,764 bytes	4,446,727 bytes	105,389,599 bytes
Size (zipped OME-TIFF with LZW)	264,875 bytes	4,937,246 bytes	116,418,287 bytes
Size (7-zipped OME-TIFF)	209,593 bytes	3,891,846 bytes	93,939,055 bytes
Size (7-zipped OME-TIFF with LZW)	264,292 bytes	4,950,897 bytes	116,567,097 bytes

Efficiency of planar access

The following table compiles our results with average plane size computed from the numbers above, and briefly summarizes each format's ability to efficiently access individual image planes. We have not performed benchmarks involving individual planar access for each format—mostly because for many of these formats (especially zip, gzip and 7-zip) it is quite impractical to attempt efficient access to individual planes.

Format	Average plane size	Efficiency of access to individual planes
Raw pixels only	Worst – 262,144 bytes	Best – Pixels can be ripped directly from disk.
OME-TIFF	Worst – 262,645 bytes	Great – IFDs identify planar offsets.
OME-TIFF+LZW	Good – 137,238 bytes	Good – The plane must be decoded from LZW, but IFDs identify planar offsets. With clever threading, each plane can be decoded while the next is being read from disk.
OME-XML	Poor – 164,942 bytes	Good – The plane must be decoded from base64, but the BinData Length attributes can be used to derive offsets without scanning the entire file. With clever threading, each plane can be decoded while the next is being read from disk.
OME-TIFF, 7-zipped	Best – 108,692 bytes	Poor – The appropriate OME-TIFF file must be uncompressed from the archive.
OME-TIFF, zipped	Great – 122,031 bytes	Poor – The appropriate OME-TIFF file must be uncompressed from the archive.
OME-TIFF+LZW, zipped	Good – 134,834 bytes	Poor – The appropriate OME-TIFF file must be uncompressed from the archive, and the plane must be decoded from LZW.
OME-TIFF+LZW, 7-zipped	Good – 135,014 bytes	Poor – The appropriate OME-TIFF file must be uncompressed from the archive, and the plane must be decoded from LZW.
OME-XML, gzipped	Great – 124,763 bytes	Worst – Entire dataset must be uncompressed, then the plane must be decoded from base64.
OME-XML, zipped	Great – 124,764 bytes	Worst – Entire dataset must be uncompressed, then the plane must be decoded from base64.
OME-XML, 7-zipped	Great – 125,830 bytes	Worst – Entire dataset must be uncompressed, then the plane must be decoded from base64.

The performance penalty for accessing individual image planes from externally compressed formats (zip, gzip, and 7-zip) is high, since the data must be decompressed. There is little penalty for accessing them from uncompressed OME-XML or OME-TIFF—with OME-XML, file readers can build a list of offsets by skipping past the bulk of the BinData characters according to the Length attribute values, and with OME-TIFF, file readers can seek to the offsets indicated in the IFD entries.

Accessing them from an uncompressed OME-TIFF file, however, is efficient. In addition, the TIFF format maintains compatibility with the multitude of existing software that works with single- and multi-page TIFF files.

Space required on disk

As shown in the table above, and the chart *OME-TIFF space used*, our figures indicate that the most efficient format for space is OME-TIFF compressed with the 7-zip utility. Also good are gzipped OME-XML and zipped OME-TIFF.

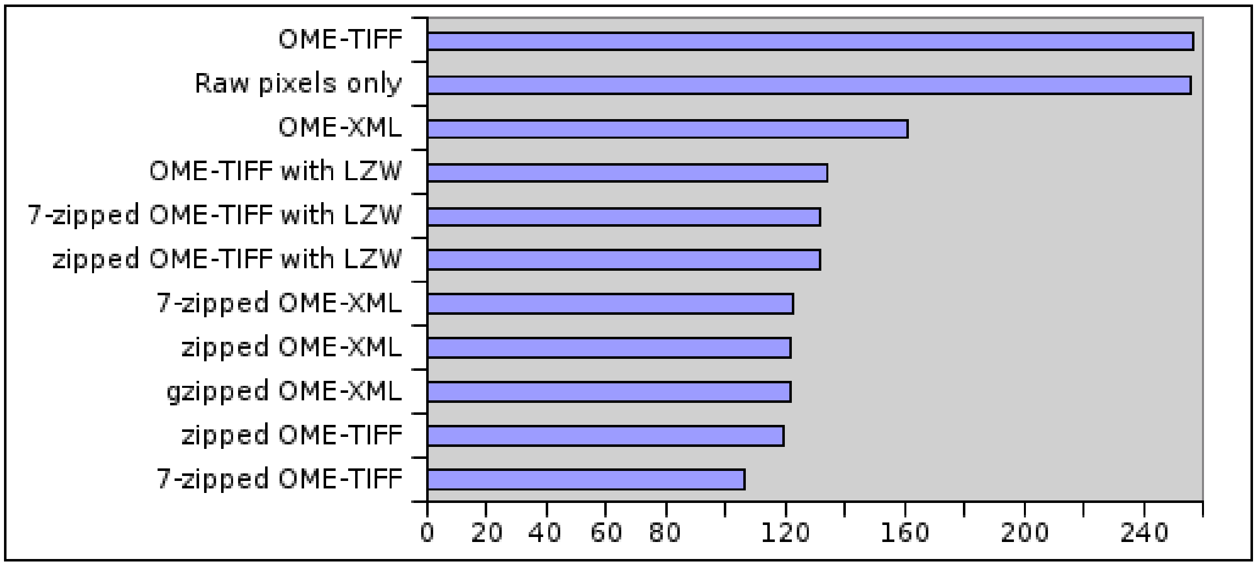


Fig. 1: OME-TIFF space used

Uncompressed OME-TIFF format, while the least space-efficient, provides its own advantages: it is highly compatible and provides efficient access to individual image planes.

OME-TIFF with LZW often performs nearly as well as the externally compressed formats (zip, gzip and 7-zip), without the performance penalty of searching through a compressed archive. However, LZW is noticeably less efficient to decode than uncompressed TIFF is, and LZW is an additional requirement for client software—it may be that some software supports uncompressed multi-page TIFF, but not LZW. Even more unfortunate, the 7-zip algorithm appears to perform less well on LZW-compressed OME-TIFFs than on uncompressed OME-TIFFs.

Recommendations

In conclusion, we highlight the following formats as most useful, depending on your circumstances:

- **Zipped OME-TIFF** – cuts down on file size while retaining the underlying compatibility of OME-TIFF. Use zipped OME-TIFF for wide distribution of data to colleagues. Zip is a ubiquitous compression format, decodable on all major operating systems. The downside is that it typically does not compress as well as 7-zip does.
- **7-zipped OME-TIFF** – minimizes file size. As the most space-efficient format, use 7-zipped OME-TIFF for archival purposes, for transport from one OME database to another, and possibly for online distribution if space is a major concern and your target audience is computer-savvy enough to install 7-zip. The only downsides are that 7-zip is less ubiquitous than zip, and compressing a dataset with 7-zip takes longer than doing so with zip.
- **OME-TIFF with LZW** – provides space advantages nearly as good as the externally compressed formats (zip, gzip, 7-zip) without sacrificing the accessibility of TIFF, for the most part. The downsides are that LZW takes somewhat longer to process than uncompressed TIFF, some software does not support LZW compression, and OME-TIFF with LZW does not shrink as much as uncompressed OME-TIFF does with external compression techniques.
- **OME-TIFF** – maximizes compatibility. When actively working with data, storing it in uncompressed OME-TIFF format provides many options for efficient analysis and visualization. The downside is that the data takes up more space on disk.

- **OME-XML** – provides metadata in a directly human-readable form. In addition, OME-XML maximizes compatibility with XML software. The downside is that very few image software packages support OME-XML.

5.1.3 Sample image files

Sample OME-XML and OME-TIFF files can be found here: Image downloads resource.

Each schema version's samples are located in a folder named after the schema's date e.g. folder https://downloads.openmicroscopy.org/images/OME-TIFF/2016-06/ has samples using schema version https://www.openmicroscopy.org/Schemas/OME/2016-06.

See the *OME-TIFF sample data* section for more information about the sample TIFF files. The OME-XML files contained in the https://downloads.openmicroscopy.org/images/OME-XML folder are named to reflect the aspects of the Data Model they demonstrate so they should be self-explanatory.

5.1.4 IDs and LSIDs in OME-XML

The ID types used throughout the OME-XML model are designed to support identifiers in two forms. Where possible, the full LSID format should be used. If an LSID resolver is unavailable, an internal-only form may be used.

An LSID is a Life Science Identifier. It is a Uniform Resource Name standard, designed to allow the unique identifying of life sciences resources across the World Wide Web in line with the Semantic Web concept. It was designed to allow the naming or identifying of data and associated metadata that can be stored in multiple, distributed data stores.

For further information see https://en.wikipedia.org/wiki/LSID.

The format of a valid LSID is:

```
URN:LSID:<Authority>:<Namespace>:<ObjectID>[:<Version>]
```

In OME-XML this is implemented as

```
urn:lsid:<domain-name>:<element-name>:<uniqueID>
```

The uniqueID can be any non-whitespace characters. The domain-name is any standard character (including Unicode) with dash and dot. It must contain at least one dot. The version block is not required but will be accepted if present.

The LSID specification defines the first three portions as '"case-insensitive"', that is `URN:LSID:<Authority>`. The remaining portion is '"case-sensitive"'. In OME-XML however, we assume '"lower case"' for the first two portions `urn:lsid`, for `<domain-name>` any case is acceptable but lower case is recommended for consistency. The remaining portion is case-sensitive.

The shorter internal only form is:

```
<element-name>:<uniqueID>
```

The formats are enforced by the regular expressions defined in the schema document e.g. a sample regular expression for a Project ID is

```
(urn:lsid:([\w-\.]+\.[\w-\.]+)+:Project:\S+)|(Project:\S+)
```

Note: The regex parser in XSD is slightly non standard and assumes that the pattern is always meant to start at the beginning of the line and finish at the end of the line, this means that !^ and $ are not necessary.

The simple regular expressions used provide a first level of validation but it is possible to produce an invalid LSID that will be accepted by the regex. As a tradeoff between complexity and usability, the domain-name check is quite lax e.g. it will accept www.ome-xml..org as valid despite the double dot.

Sample IDs

Valid

```
urn:lsid:sample.ome-xml.org:Project:1234
Project:1234
```

Invalid

```
sample.ome-xml.org:Project:1234 (domain but no protocol name)
1234 (no element name)
```

5.1.5 The OME system of units

Implementation

The *ome.xsd* schema file defines new type enumerations: UnitsLength, UnitsTime, UnitsPressure, UnitsAngle, UnitsTemperature, UnitsElectricPotential, UnitsPower and UnitsFrequency. These types are used throughout the model. Each new unit attribute is in a pair with an existing attribute, e.g. PinholeSize and PinholeSizeUnit. The **xsd-fu** code generator then processes these attribute pairs into a single unit aware attribute in the output java classes. The java classes make use of the types from the new *ome.units.** classes.

The classes in *ome.units* are inspired by but not a direct implementation of the specification for a Units of Measurement API.

These classes have been constructed to allow for future wrapping of the Units of Measurement API implementation developed at https://www.eclipse.org/uomo/ if future work on that project provides a more complete implementation.

See also:

OMERO developer documentation on units for further details of how this aspect of the Data Model is implemented in OMERO.

Default and Alternate OME-XML samples demonstrating the use of instrument units.

OME system of measurements

A system of measurements is defined for:

- *Angle*
- *Electric potential (commonly called voltage)*
- *Frequency*
- *Length*
- *Power*

Angle

Is based on the following unit:

- **rad** - radian (the SI unit of angle)

And from this are derived the following units:

- **deg** - degree
- **gon** - gradian

All angle units are freely convertible.

Electric potential (commonly called voltage)

Is based on the following unit:

- **V** - volt (the SI unit of electric potential)

And from this are derived the following units:

- All the SI range for volt from 10^{24} yotta (**YV**) to 10^{-24} yocto (**yV**)

All electric potential units are freely convertible.

Frequency

Is based on the following unit:

- **Hz** - hertz (the SI unit of frequency)

And from this are derived the following units:

- All the SI range for hertz from 10^{24} yotta (**YHz**) to 10^{-24} yocto (**yHz**)

All frequency units are freely convertible.

Length

Is based on the following units:

- **m** - meter (the si unit of length)
- **pixel** - a length measured in terms of the image pixel size
- **reference frame** - a length measured in terms of an arbitrary unit based on the equipment used

And from these are derived the following units:

- All the SI range for meter from 10^{24} yotta (**Ym**) to 10^{-24} yocto (**ym**)
- **Å** - ångströms

- **in** - inch (the Imperial/US unit of length)
- **thou** - thou (or mil, 1/1000 of an inch)
- **li** - line (1/12 of an inch)
- **in** - inch
- **ft** - foot
- **yd** - yard
- **mi** - terrestrial mile
- **ua** - astronomical unit - *The official term is ua as the SI standard assigned AU to absorbance unit.*
- **ly** - light year
- **pc** - parsec
- **pt** - typography point - *This unit should be limited to font sizes.*

All units are freely convertible **except** for **pixel** and **reference frame**, both of which are of unspecified length.

Power

Is based on the following unit:

- **W** - watt (the SI unit of power)

And from this are derived the following units:

- All the SI range for watt from 10^{24} yotta (**YW**) to 10^{-24} yocto (**yW**)

All power units are freely convertible.

Pressure

Is based on the following unit:

- **Pa** - pascal (the si unit of pressure)

And from this are derived the following units:

- All the SI range for pascal from 10^{24} yotta (**YPa**) to 10^{-24} yocto (**yPa**)
- **Mbar** - mega bar
- **kbar** - kilo bar
- **dbar** - deci bar
- **cbar** - centi bar
- **mbar** - milli bar
- **atm** - standard atmosphere
- **psi** - pounds per square inch
- **Torr** - torr
- **mTorr** - milli torr
- **mm Hg** - millimetre of mercury

All pressure units are freely convertible.

Temperature

Is based on the following unit:

- **K** - kelvin (the SI unit of temperature)

And from this are derived the following units:

- °**C** - degree Celsius
- °**F** - degree Fahrenheit
- °**R** - degree Rankine

The degree sign and word was dropped from kelvin in 1968. https://en.wikipedia.org/wiki/Kelvin

All temperature units are freely convertible.

Time

Is based on the following unit:

- **s** - second (the SI unit of time)

And from this are derived the following units:

- All the SI range for second from 10^{24} yotta (Ys) to 10^{-24} yocto (ys)
- **min** - minute
- **h** - hour
- **d** - day

All time units are freely convertible.

General points

Unit abbreviations

The string used for each unit is the standard abbreviation for that unit. In a few cases these do not seem obvious but the set of abbreviations has been chosen by the scientific community to avoid abbreviation clashes.

Unit names

The name and spelling used for each unit in long form is not defined by the scientific community, rather the abbreviation is standardised. For example, the unit of length with the symbol **m** is equally valid written as meter, metre, metr, or metro.

The SI range of prefixes

The following unit abbreviation are defined:

- **Y** - 10^{24} - yotta
- **Z** - 10^{21} - zetta
- **E** - 10^{18} - exa
- **P** - 10^{15} - peta
- **T** - 10^{12} - tera
- **G** - 10^{9} - giga
- **M** - 10^{6} - mega
- **k** - 10^{3} - kilo
- **h** - 10^{2} - hecto
- **da** - 10^{1} - deca
- **d** - 10^{-1} - deci
- **c** - 10^{-2} - centi
- **m** - 10^{-3} - milli
- **µ** - 10^{-6} - micro
- **n** - 10^{-9} - nano
- **p** - 10^{-12} - pico
- **f** - 10^{-15} - femto
- **a** - 10^{-18} - atto
- **z** - 10^{-21} - zepto
- **y** - 10^{-24} - yocto

The OME Data Model is now decoupled from the Bio-Formats code repository and available as a stand-alone ome-model GitHub repository.

There are sample files, along with an explanation of their structure, on the *Sample image files* page.

5.1.6 OME Model development process

The Model development process is currently being revised but we always aim to keep the community informed of major and breaking changes in advance. See the Contributing Developer documentation for further details on updating and publishing the schema.

5.1.7 Working with the OME Data Model

The *Model Overview collection of diagrams* shows the structure and connections between different parts of the OME Model. Generated documentation for the current version of the entire Schema is also available.

Individual parts of the model are covered in more detail in the following sections:

- *Filter And FilterSet*
- *Screen Plate Well* - our HCS solution
- *Structured Annotations*
- *Regions of Interest (ROIs)*

Support for additional dimensions is also covered:

- *6D, 7D and 8D Storage*

Map Annotations, storing 'key-value pairs', are a type of Structured Annotation which were introduced in the *Changes for January 2015*. Further information is available in the OMERO developer documentation on Key-value pairs and a sample OME-XML file is also available.

Legacy solutions for tiled images and Single (or Selective) Plane Illumination Microscopy (also known as Light Sheet Microscopy) are detailed in the following sections for reference but **both these methods have been superseded by the above**:

- *SPIM Initial Support*
- *Tiled Images*

The use of IDs and Life Science Identifiers is explained in this section:

- *IDs and LSIDs in OME-XML*

The system of units used by the model is covered in this section:

- *The OME system of units*

The *Schema versions* section shows the Model changes with each release, helpful for those working with several versions of the OME Model, for example to support the loading/saving of a variety of files.

- The **current major release** - see *Changes For June 2016*.

For further information, see the OME Data Model section in the OMERO developer documentation.

5.1.8 Working with OME-XML

In some cases, it is useful to extract specific parameters or tweak certain values in a dataset's OME-XML metadata block. Further guidance on *Using OME-XML schema elements* is available, but below is a brief example of the OMEXMLMetadata class (which implements the MetadataStore and MetadataRetrieve interfaces) to greatly simplify OME-XML-related development tasks.

The following program edits the "image name" metadata value for the file given on the command line. It requires the Bio-Formats and *OME-XML Java* libraries.

EditImageName.java

As in the ConvertToOmeTiff.java example in *OME-TIFF example source code for common operations*, we attach an OME-XML MetadataStore object to the reader to extract OME-XML metadata from the input file. We obtain the current image name (if any) by calling the `omexmlMeta.getImageName(0)` method. The zero indicates the Image within the OME-XML block we are interested in; in this case, we are asking for the name of the first Image.

After updating the name somehow (in our case, reversing the string), we write the updated name back into the metadata structure via a call to `omexmlMeta.setImageName(name, 0)`. Once again the zero indicates that we wish to update the first Image.

Lastly, the loci.formats.services.OMEXMLService class contains a number of useful methods for working with Bio-Formats metadata objects (i.e. MetadataStore and MetadataRetrieve implementations), including the getOMEXML method for easily extracting an OME-XML string from a MetadataRetrieve object (which we utilize above), as well as the convertMetadata method for transcoding between metadata object implementations. You can obtain an OMEXMLService object as follows:

```
ServiceFactory factory = new ServiceFactory();
OMEXMLService service = factory.getInstance(OMEXMLService.class);
```

5.1.9 Additional tools

The **`xsd-fu`** code generator digests the OME data model schema and produces an object oriented Java infrastructure to ease working with an XML DOM tree.

CHAPTER

SIX

THE DATA MODEL IN DETAIL

6.1 Schema Generation information

Generated documentation for the current version of the entire Schema is also available.

6.1.1 Current Data Model overview

The diagrams below illustrate some aspects of the model and further details are given on the following pages. Generated documentation for the current version of the entire Schema is also available.

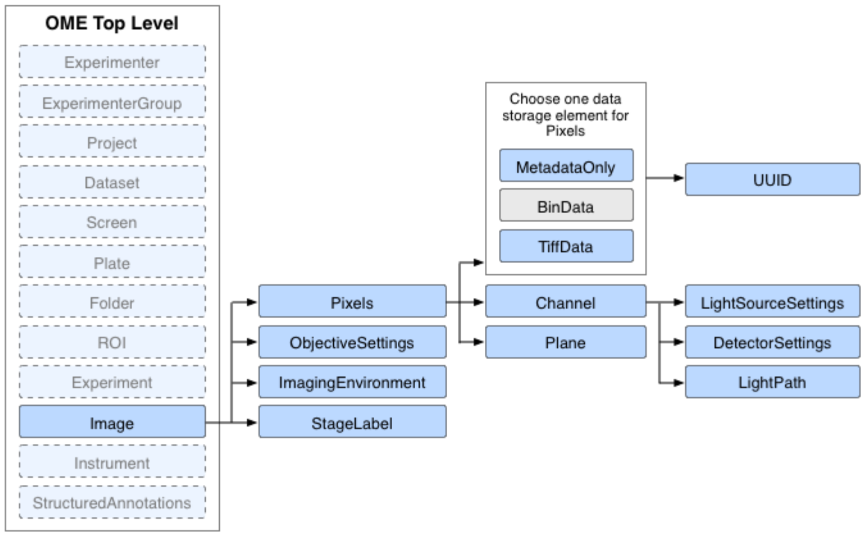


Fig. 1: Image branch of the OME Schema

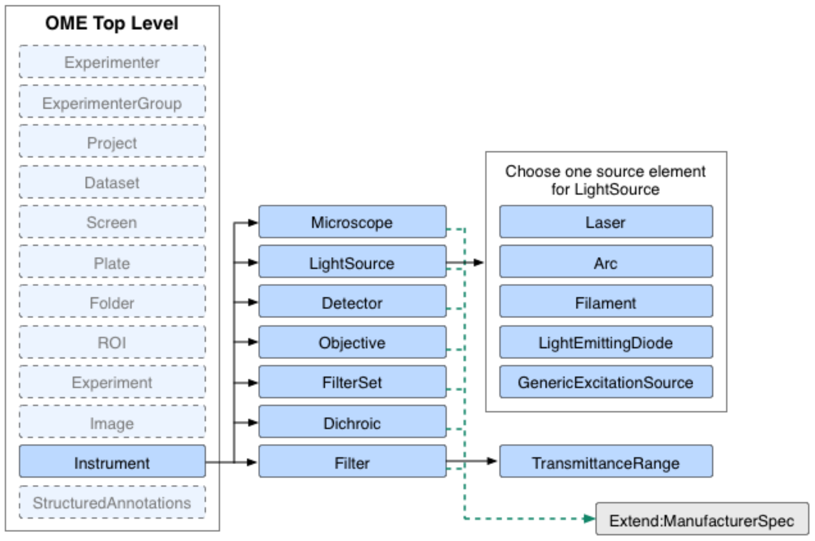


Fig. 2: Instrument branch of the OME Schema

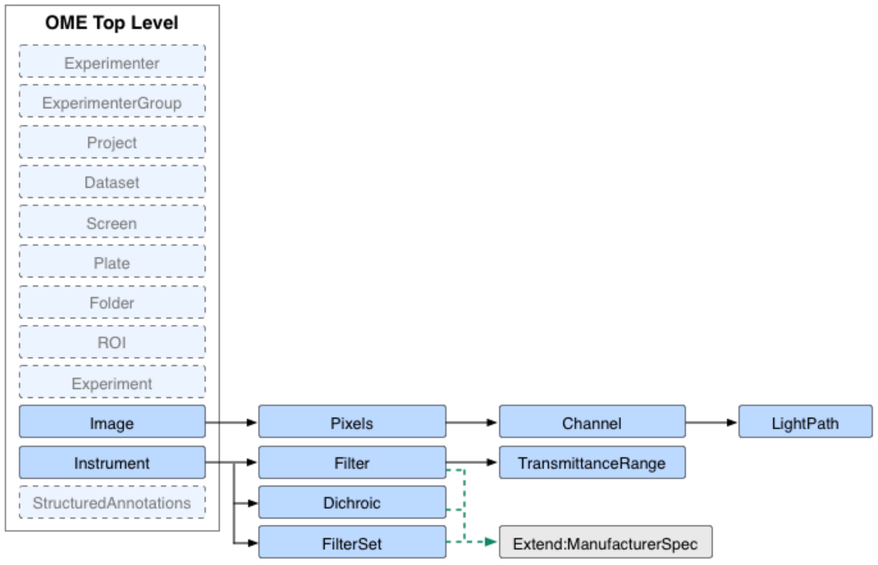


Fig. 3: Location of Filters and Light Paths in the OME Model

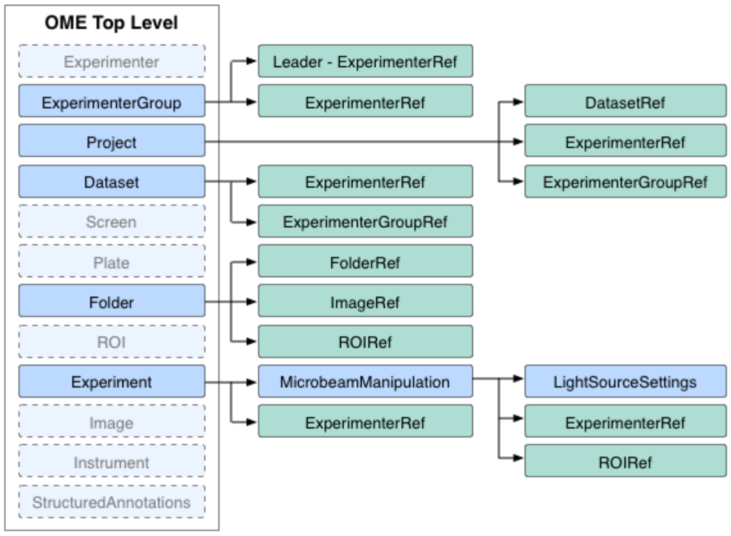


Fig. 4: Organizational structures (Project, Dataset, Group, Experiment, Experimenter) of the OME Model

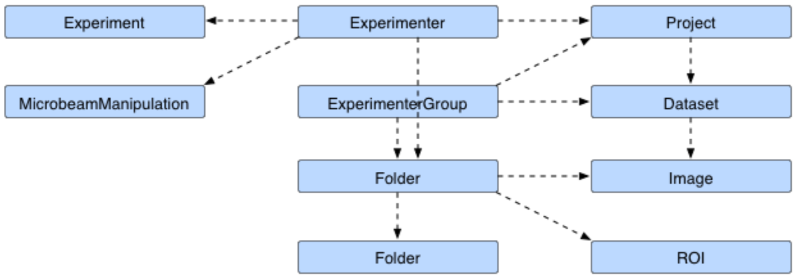


Fig. 5: Ownership relations for Project, Dataset and Experimenter

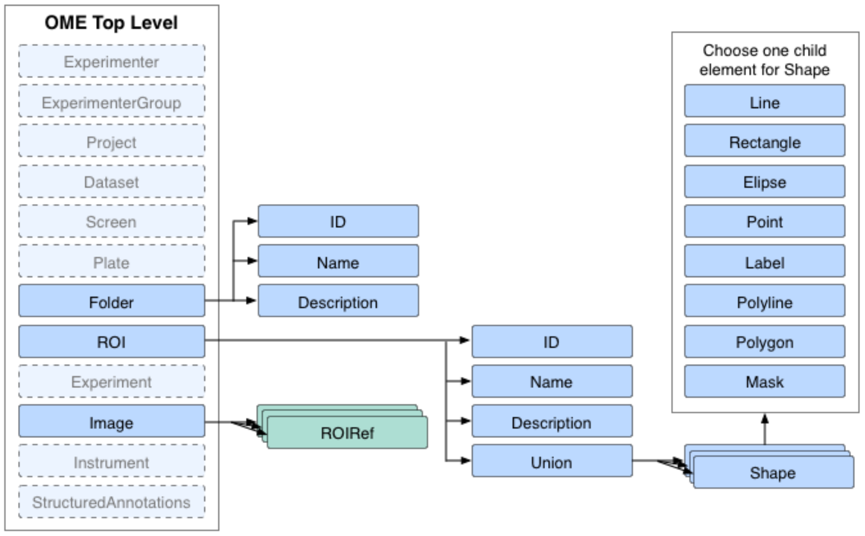


Fig. 6: Region Of Interest branch of the OME Model

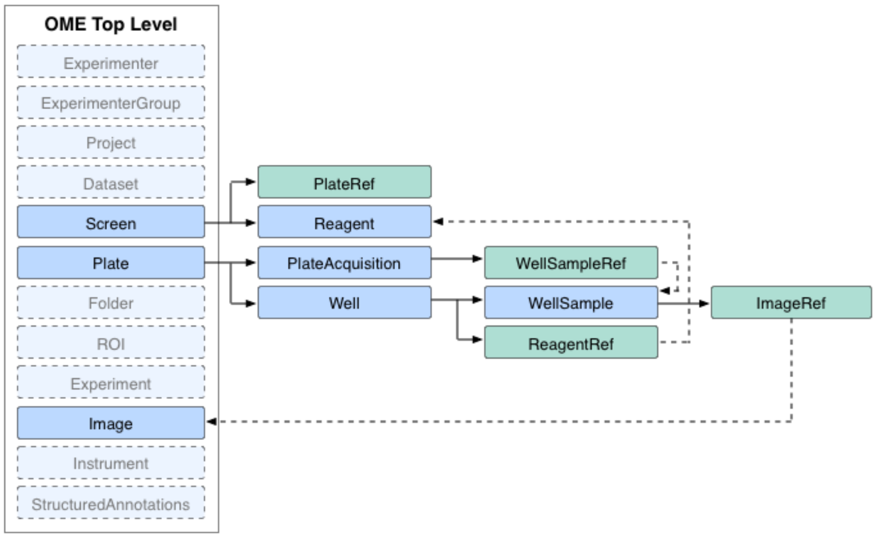


Fig. 7: HCS structures (Screen, Plate and Well) of the OME Model

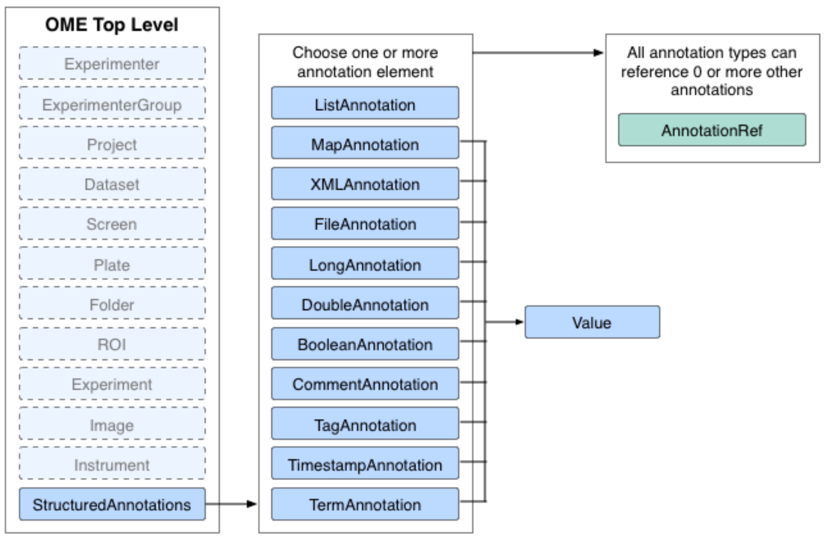


Fig. 8: StructuredAnnotation branch of the OME Model

6.1.2 Filter and FilterSet

The filter model changed with the April 2010 release of the OME Schema. This page provides a description of the new structures used. The new structures are designed to work for Fluorescence Microscopy but are flexible enough it can also work with filter configurations for most other forms of microscopy.

Supported Objects

The schema supports the following objects:

Dichroic
The dichroic mirror or dichromatic beamsplitter. This is used for the primary dichroic of the instrument. Other dichroics in the excitation or emission path can be modeled with the Filter object.

Filter
This object is used to represent any type of excitation filter and emission/barrier filter. It contains details of the transmittance of the filter. It can also be used to model a dichroic mirror or dichromatic beamsplitter in the excitation or emission path.

FilterSet
A FilterSet belongs to an Instrument in an OME-XML file and has alongside it any Filters and Dichroics it refers to. It is designed to represent a collection of filters available on a certain instrument. It can also represent a filter cube or filter block. To represent the set of filters used to collect an image, LightPath is a better choice. FilterSet may contain zero or more excitation Filters, one Dichroic, and zero or more emission Filters. No ordering of Filters is implied. Filters and Dichroics can be reused across multiple FilterSets. It will be normal to have a single Dichroic that is usable with several Filter combinations.

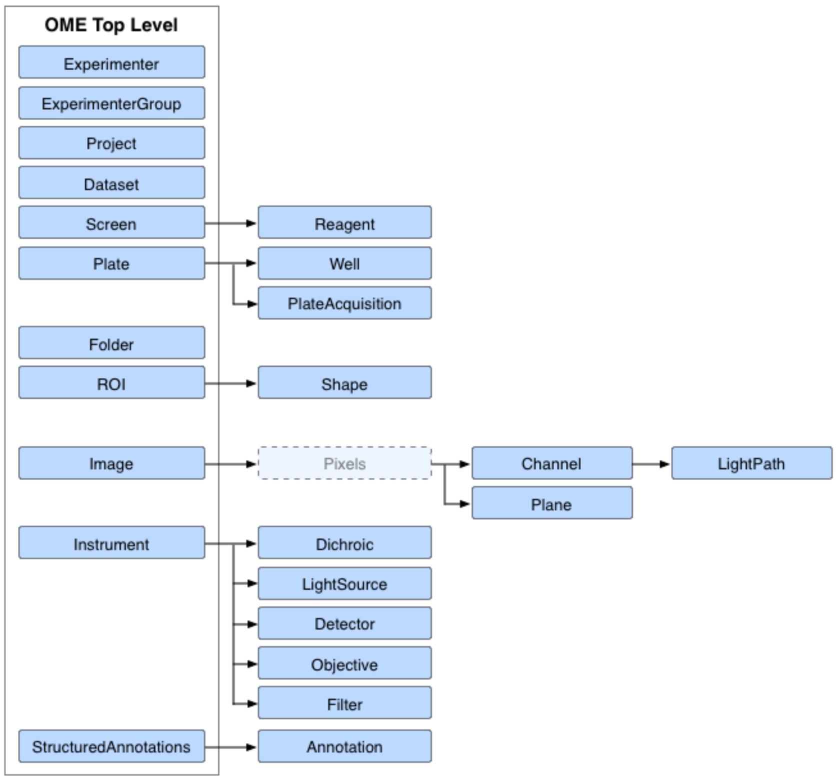


Fig. 9: All points in the OME Model that can be Annotated

FilterSetRef
These are used to attach a specific FilterSet from an Instrument to a Channel within an Image. They allow the FilterSet to be defined once in the instrument in an OME-XML file, and then used multiple times within the images.

ManufacturerSpec
The Manufacturer spec is used by both Dichroic and Filter to store the Manufacturer, Model, SerialNumber and LotNumber of the optical element.

FilterWheel
This is a value within a Filter used to record which holder the filter is located in. Although it is named after a wheel, it could just as easily be a filter slider or other mechanism. These holders allow any of their containing Filters to be automatically selected by the microscope control software.

LightPath
The LightPath object is a child of Channel and is used to represent the collection of Dichroic and Filters used to create the Channel. FilterSet is on an Instrument granularity, LightPath on a Channel of an Image.

Filters and Dichroics in the optical path

Key for figure *Sample instrument light path*

1. The source produces a beam of light.
2. A small range of wavelengths are passed through the filters in the excitation path.
3. The light is selectively reflected by the dichroic mirror to illuminate the sample.
4. The fluorphore in the sample absorbs the light from 3 and re-emits it on a new wavelength. Some of this re-emitted light travels towards the dichroic mirror.
5. The light on the new wavelength is selectively transmitted by the dichroic mirror.
6. The wavelength of detected light is further narrowed and stray reflections reduced by the filters in the emission path.
7. The light can be optionally split by a dichroic in the emission path to go to two, or more, detectors. This dichroic is modelled as a Filter of type "Dichroic".
8. The split light beam can travel directly from the dichroic to a detector.
9. The beam may be further filtered before reaching a detector.

Filter data structure in OME-XML

The Filter data elements are part of the central OME Schema. The filters are part of the Instrument block. Each instrument can contain zero or more filters, dichroics, and filter sets. Each filter set uses unique IDs to refer to the filters and dichroic it contains. If there are any additional filters used with the Channels within an image, then those filters are also defined at the instrument level, and referenced from the channel with a LightPath element using the unique ID. The Filters and Dichroics are both extensions of the ManufacturerSpec element so store this common data. A filter also contains one TransmittanceRange element that describes its optical characteristics.

The FilterSet contains three kinds of references, one to a dichroic ID called DichroicRef, and two sets to filter IDs called ExcitationFilterRef and EmissionFilterRef. Each of the three kinds are optional. There is no distinction between a filter designed for emission or excitation at the Filter level. The distinction is made within the FilterSet (or LightPath) by the assignment of the filter ID reference to either a ExcitationFilterRef or EmissionFilterRef. Not all the objects in a FilterSet need to be used at the same time, and there is no ordering implied within the groups of references.

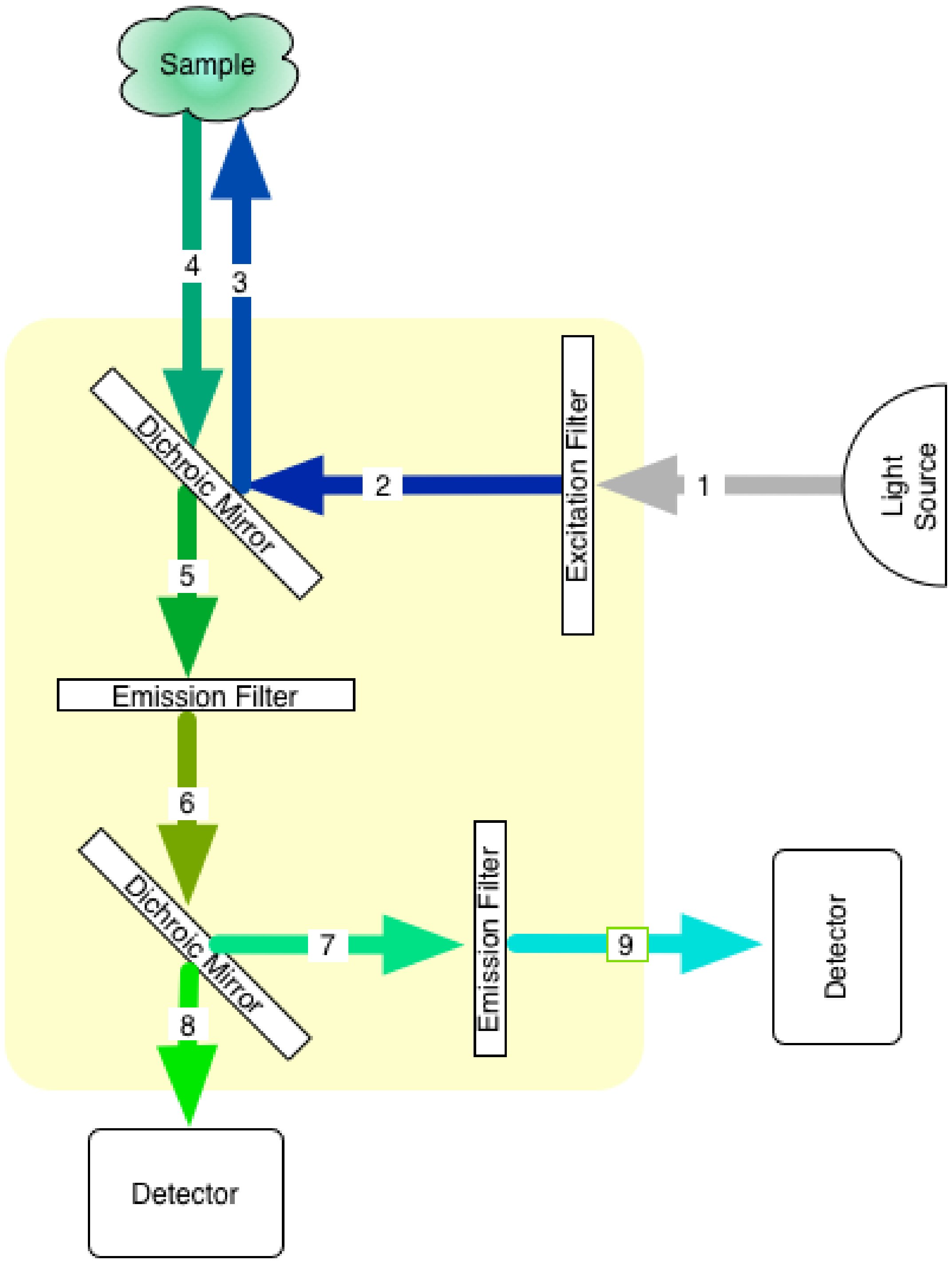


Fig. 10: Sample instrument light path

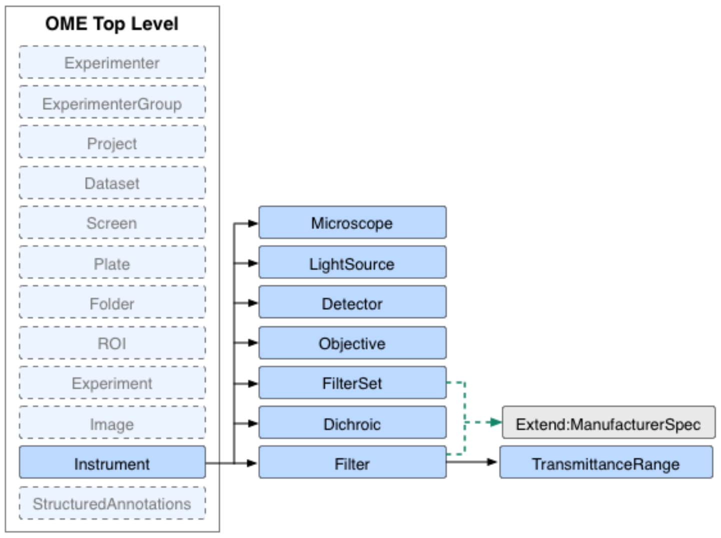


Fig. 11: Where FilterSet sits in the OME Schema

The Dichroic object simply contains its unique ID and the values it gets from extending ManufacturerSpec. In ManufacturerSpec, each object can have a SerialNumber and/or a LotNumber. The LotNumber is more commonly used in Filters and Dichroics so the batch of manufacture is recorded.

The Filter object has the values it gets by extending ManufacturerSpec and a unique ID the same as the Dichroic. It also has a Type and a FilterWheel value. In addition to these extra values, it contains one TransmittanceRange object. The TransmittanceRange describes the optical characteristic of the filter and has values for the CutIn and CutOut, along with the CutInTolerance and CutOutTolerance, all expressed in nanometers. There is also the Transmittance of the filter expressed as a percentage fraction.

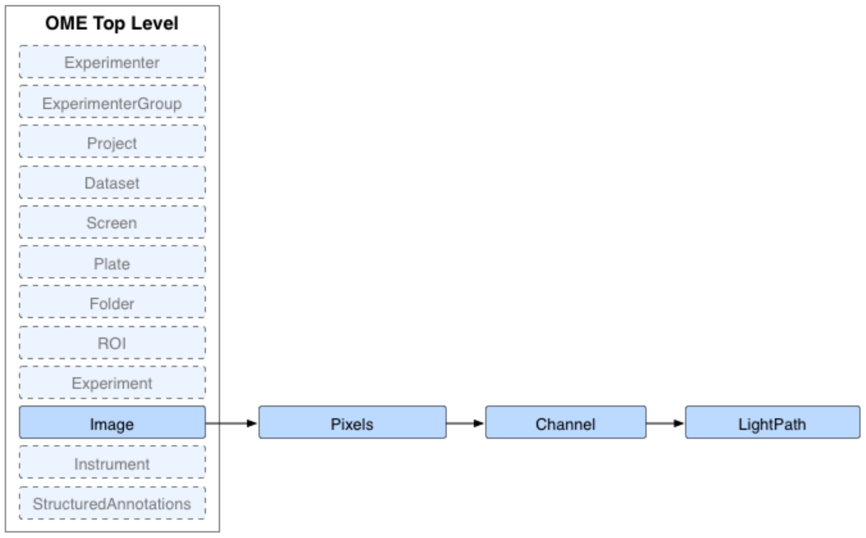


Fig. 12: Where LightPath sits in the OME Schema

The LightPath contains three kinds of references, one to a dichroic ID called DichroicRef, and two sets to filter IDs called ExcitationFilterRef and EmissionFilterRef. Each of the three kinds are optional. There is no distinction between a filter designed for emission or excitation at the filter level. The distinction is made within the LightPath by the assignment of the filter ID reference to either an ExcitationFilterRef or EmissionFilterRef. The key difference between LightPath and FilterSet is that ALL the filters in a LightPath have been used and their order is specified.

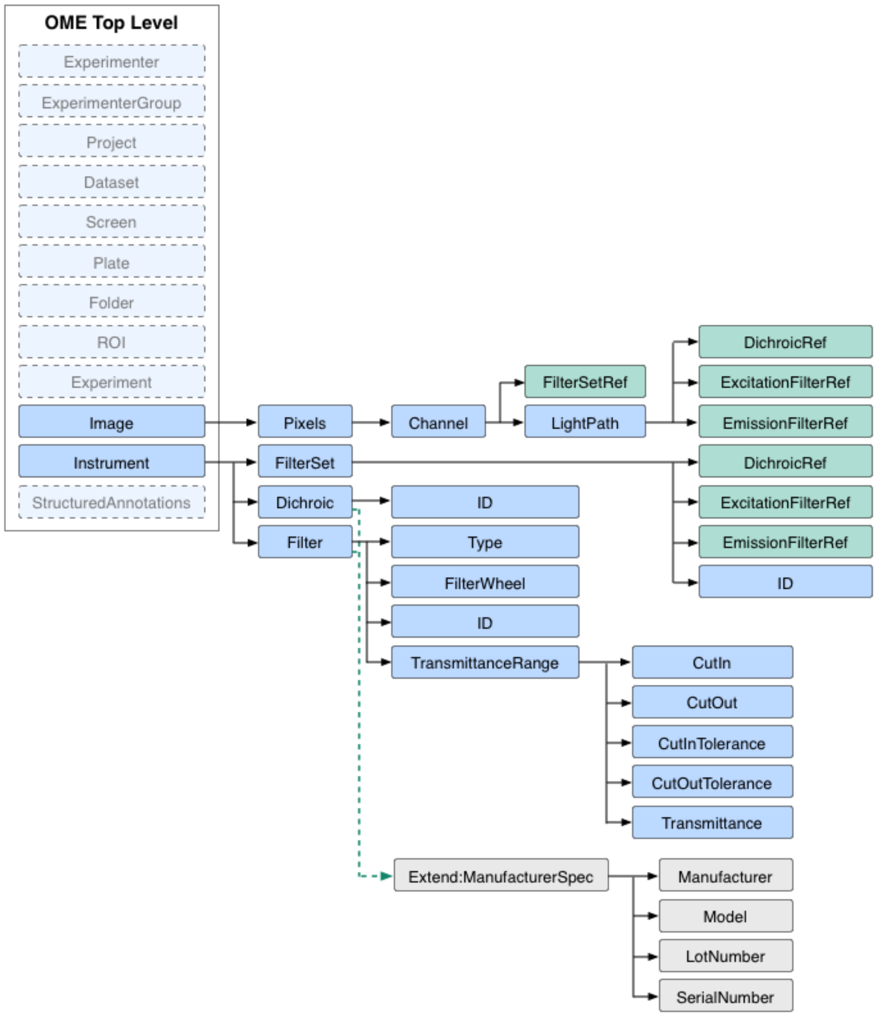


Fig. 13: Attributes within the Filter objects

Sample pieces of .ome.xml files

Note: The sample sections below are taken from https://downloads.openmicroscopy.org/images/OME-XML/2016-06/filter.ome.xml

This first example shows the Instrument side - the Filter/Dichroic/FilterSet structure of the resulting xml.

- It defines six Filters; five are standard filters, the sixth is a dichroic used as a filter.
- It defines two Dichroics that can represent the primary Dichroic in an instrument.
- FilterSet:1 contains Filter:1 and/or Filter:2 for excitation, Dichroic:1 and Filter:3 and/or Filter:4 for emission.

```
<FilterSet ID="FilterSet:1" Manufacturer="Ink Inc." Model="Mk 3"
    LotNumber="K753">
    <ExcitationFilterRef ID="Filter:1"/>
    <ExcitationFilterRef ID="Filter:2"/>
    <ExcitationFilterRef ID="Filter:3"/>
    <ExcitationFilterRef ID="Filter:4"/>
    <DichroicRef ID="Dichroic:1"/>
    <EmissionFilterRef ID="Filter:5"/>
    <EmissionFilterRef ID="Filter:6"/>
</FilterSet>
<FilterSet ID="FilterSet:2" Manufacturer="Ink Inc." Model="Mk 3"
    LotNumber="K753"/>
<Filter ID="Filter:1" Manufacturer="Ink Inc." Model="Medium 490"
    LotNumber="J23" Type="BandPass" FilterWheel="Disk 7">
    <TransmittanceRange Transmittance="0.80" CutIn="450" CutOut="530"/>
</Filter>
<Filter ID="Filter:2" Manufacturer="Ink Inc." Model="Medium 520"
    LotNumber="J34" Type="BandPass" FilterWheel="Disk 7">
    <TransmittanceRange Transmittance="0.75" CutIn="500" CutOut="570"/>
</Filter>
<Filter ID="Filter:3" Manufacturer="Ink Inc." Model="Medium 580"
    LotNumber="J12" Type="BandPass" FilterWheel="Disk 7">
    <TransmittanceRange Transmittance="0.85" CutIn="550" CutOut="620"/>
</Filter>
<Filter ID="Filter:4" Manufacturer="Ink Inc." Model="Medium 630"
    LotNumber="J09" Type="BandPass" FilterWheel="Disk 7">
    <TransmittanceRange Transmittance="0.90" CutIn="590" CutOut="680"/>
</Filter>
<Filter ID="Filter:5" Manufacturer="Ink Inc." Model="Output 724"
    LotNumber="J34" Type="MultiPass">
    <TransmittanceRange Transmittance="0.75" CutIn="500" CutOut="570"/>
</Filter>
<Filter ID="Filter:6" Manufacturer="Ink Inc." Model="Medium 762"
    LotNumber="J12" Type="MultiPass">
    <TransmittanceRange Transmittance="0.85" CutIn="550" CutOut="620"/>
</Filter>
<Filter ID="Filter:7" Manufacturer="Ink Inc." Model="Medium 672"
    LotNumber="J09" Type="ShortPass">
    <TransmittanceRange Transmittance="0.90" CutIn="590" CutOut="680"/>
</Filter>
<Filter ID="Filter:Dichroic:2" Model="MirrorBlock Mk II" LotNumber="M538"
    Type="Dichroic"/>
<Dichroic ID="Dichroic:1" Model="HFT 405/488/543/633"/>
<Dichroic ID="Dichroic:3" Model="MirrorBlock MK II" LotNumber="M539"/>
```

This second example shows the Image side - the LightPath structure of the resulting xml.

- Channel:1 defines a LightPath that uses Filter:1 for excitation, Dichroic:1, then Filter:2 and Filter:6 for Emission in that order.
- Channel:2 defines a LightPath that uses Filter:1 for excitation, Dichroic:1, then Filter:2, Filter:6 and Filter:5 for Emission in that order.
- Channel:3 references FilterSet:1, from this we do not know which of the filters in that FilterSet were used and in which order.

- Dichroic:2 though defined above is not used by this Image.

```
<Image ID="Image:0" Name="405100percentsetting">
    <OME:AcquisitionDate>2008-06-19T00:39:00</OME:AcquisitionDate>
    <Description>Sample Image</Description>
    <InstrumentRef ID="Instrument:0"/>
    <ObjectiveSettings ID="Objective:0:0"/>
    <OME:Pixels ID="Pixels:1" DimensionOrder="XYCTZ" Type="int16"
        SizeX="128" SizeY="128" SizeZ="1" SizeC="2" SizeT="1">
        <Channel ID="Channel:1">
            <LightPath>
                <!--  ordered collection  -->
                <ExcitationFilterRef ID="Filter:1"/>
                <ExcitationFilterRef ID="Filter:Dichroic:2"/>
                <DichroicRef ID="Dichroic:1"/>
                <EmissionFilterRef ID="Filter:5"/>
            </LightPath>
        </Channel>
        <Channel ID="Channel:2">
            <FilterSetRef ID="FilterSet:2"/>
            <LightPath>
                <EmissionFilterRef ID="Filter:6"/>
            </LightPath>
        </Channel>
        <MetadataOnly/>
    </OME:Pixels>
</Image>
```

6.1.3 Screen Plate Well

The Screen Plate Well (SPW) model is designed to support High Content Screening (HCS). It is aimed at providing a flexible framework to organize the images that result from such a screen and link to external systems that contain full information about the components and products used. Throughout the model, there are several **External Identifier** strings that can be attached to support this.

The top level of the SPW model has two objects that exist side by side - **Screen** and **Plate**. It is important to stress that **Plate** is not a child of **Screen**, they are equals. This is necessary to cater for the fact that while a **Screen** can contain many **Plate**s, a **Plate** can also be used in more than one **Screen**. This allows less common scenarios where, for example, there are two screens against two different reagent sets but only one set of control plates that are used in common.

Reagents are children of a **Screen** and as such are designed to be referenced from each **Screen** they are part of. It is worth covering the exact meaning of a **Reagent** within SPW. The storing of detailed information about biological reagents is beyond the scope of the OME model. There are several other systems that are designed to handle this type of information. What the **Reagent** element in SPW provides is useful descriptions and an external reagent identifier, that can be used to find detailed information about the reagent in another system. A side effect of this is that any change to an external reagent requires a new SPW reagent. This could be a change of dilution, supplier, or lot-number.

Note: Refer to OME-XML downloads to find sample files for different combinations of Screens, Plates and Wells.

Example

Two chemical reagents, Monastrol and VX680, applied to samples in various concentrations; Monastrol at 100, 300, and 900 nM concentrations, and VX680 at 300, 900, 2700 nM concentrations. This requires a total of 6 reagents to be defined in the OME file:

```
<spw:Screen ...>
    ...
    <spw:Reagent ID="Reagent:001" Description="Monastrol at a 100nM concentration from␣
↪XYZ Supplier" Name="Monastrol-100nM" ReagentIdentifier="R-XYZ-Mon-100-nm-00345"/>
    <spw:Reagent ID="Reagent:002" Description="Monastrol at a 300nM concentration from␣
↪XYZ Supplier" Name="Monastrol-300nM" ReagentIdentifier="R-XYZ-Mon-300-nm-01245"/>
    <spw:Reagent ID="Reagent:003" Description="Monastrol at a 900nM concentration from␣
↪XYZ Supplier" Name="Monastrol-900nM" ReagentIdentifier="R-XYZ-Mon-900-nm-00875"/>
    <spw:Reagent ID="Reagent:004" Description="VX680 at a 300nM concentration from XYZ␣
↪Supplier" Name="VX680-300nM" ReagentIdentifier="R-XYZ-VX680-300-nm-00256"/>
    <spw:Reagent ID="Reagent:005" Description="VX680 at a 900nM concentration from XYZ␣
↪Supplier" Name="VX680-900nM" ReagentIdentifier="R-XYZ-VX680-900-nm-00257"/>
    <spw:Reagent ID="Reagent:006" Description="VX680 at a 2700nM concentration from XYZ␣
↪Supplier" Name="VX680-2700nM" ReagentIdentifier="R-XYZ-VX680-2700-nm-00258"/>
    ...
</spw:Screen>
```

Elements and Attributes

Plate

ID - used by the system to identify the plate.

Name - chosen by the user to identify the plate.

Description - a free text description.

ColumnNamingConvention - the column naming convention.

Columns - the number of columns in the plate.

ExternalIdentifier - an identifier for the plate used by an external system. This may be a barcode printed on the plate by the manufacturer.

RowNamingConvention - the row naming convention.

Rows - the number of rows in the plate.

Status - the current state of the plate in the experiment work-flow.

WellOriginX - the X position to use for the origin of the fields (individual images) taken in a well.

WellOriginY - the Y position to use for the origin of the fields (individual images) taken in a well.

Reagent

ID - used by the system to identify the reagent.

Name - a short name for the reagent.

Description - a long description for the reagent.

ReagentIdentifier - a reference to an external (to OME) representation of the Reagent. It serves as a foreign key into an external database.

Screen

ID - used by the system to identify the screen.

Name - chosen by the user to identify the screen.

Description - a long description for the screen.

ProtocolDescription - a description of the screen protocol; may contain very detailed information to reproduce some of that found in a screening database.

ProtocolIdentifier - a pointer to an externally defined protocol, usually in a screening database.

ReagentSetDescription - a description of the reagent set; may contain very detailed information to reproduce some of that found in a screening database.

ReagentSetIdentifier - a pointer to an externally defined reagent set, usually in a screening database/automation database.

Type - a human-readable identifier for the screen type e.g. RNAi, cDNA, SiRNA.

PlateAcquisition

PlateAcquisition is used to describe a single acquisition run for a plate. Since Plates are abstract, this object is used to record the set of images acquired in a single acquisition run. The images for this run are linked to PlateAcquisition through WellSample.

ID - used by the system to identify the plate acquisition.

Name - chosen by the user to identify the plate acquisition.

EndTime - time when the last image of this acquisition was collected.

MaximumFieldCount - the maximum number of fields (well samples) in any well in this PlateAcquisition.

StartTime - time when the first image of this acquisition was collected

Well

A Well is a component of the Well/Plate/Screen construct to describe screening applications. A Well has a number of WellSample elements that link to the Images collected in this well. The ReagentRef links any Reagents that were used in this Well. A Well is part of one or more Plates. The origin for the row and column identifiers is the top left corner of the plate, starting at zero.

ID - used by the system to identify the well.

Color - a marker color used to highlight the well.

Column - the column index of the well; the origin is the top left corner of the plate with the first column of cells being column zero i.e top left is (0,0).

ExternalDescription - a description of the externally defined identifier for this plate.

ExternalIdentifier - a pointer to an externally defined identifier for this plate.

Row - the row index of the well; the origin is the top left corner of the plate with the first row of wells being row zero i.e top left is (0,0).

Status - a human-readable identifier for the screening status e.g. empty, positive control, negative control, control, experimental. This string is likely to become an enumeration in future releases.

WellSample

WellSample is an individual image that has been captured within a Well.

ID - used by the system to identify the well sample.

Index - records the order of the well samples. Each index should be unique for a given plate but they do not have to be sequential.

PositionX - the X position of the field (image) within the well relative to the well origin defined on the Plate.

PositionY - the Y position of the field (image) within the well relative to the well origin defined on the Plate.

Timepoint - the time-point at which the image started to be collected.

ScreenAcquisition rename

In the previous version of the SPW Model, the acquisition information was recorded below **Screen**. During use, it was discovered that this did not best reflect the way data was collected. This difference between the structure and the collected data produced a performance bottleneck when dealing with the large structures necessary for HCS data. The decision was taken to rework the model to store the acquisition information on **Plate** instead. The structure was moved in the model to below **Plate** and renamed **PlateAcquisition**.

Index was also added to **WellSample** at this time. This records the order of the well samples and should be unique within a given **Plate**. It does not however, have to be sequential so supports the collection of sparse datasets.

SPW Schema structure

Within the OME data model **Screen**, **Plate** and **Image** are top level structures. A file can contain multiple instances of these elements.

Each **Plate** can contain multiple **Well**s, **PlateAcquisition**s, and **ScreenRef**s i.e. a plate can belong to multiple screens. Each **Well** in turn can have one (or zero) **ReagentRef**, and zero or more **WellSample**s. Each of the well samples defines the location of an image within the well, and the time at which the sample started to be collected. It also references the **Image** itself.

Each **Screen** contains multiple **Reagent**s, and **PlateRef**s i.e. a screen can use many plates. A **PlateAcquisition** is best viewed as a pass over some or all of the **Well**s in a **Plate**, usually associated with some time-point or step in a protocol. It is a collection of individual **WellSample**s across the **Plate**s collected between two time points.

The dashed arrows on the structure diagram show the inter-dependencies between the ‘’Screen” branch and the ‘’Plate” branch of the schema, as well as the ultimate link to ‘’Image”.

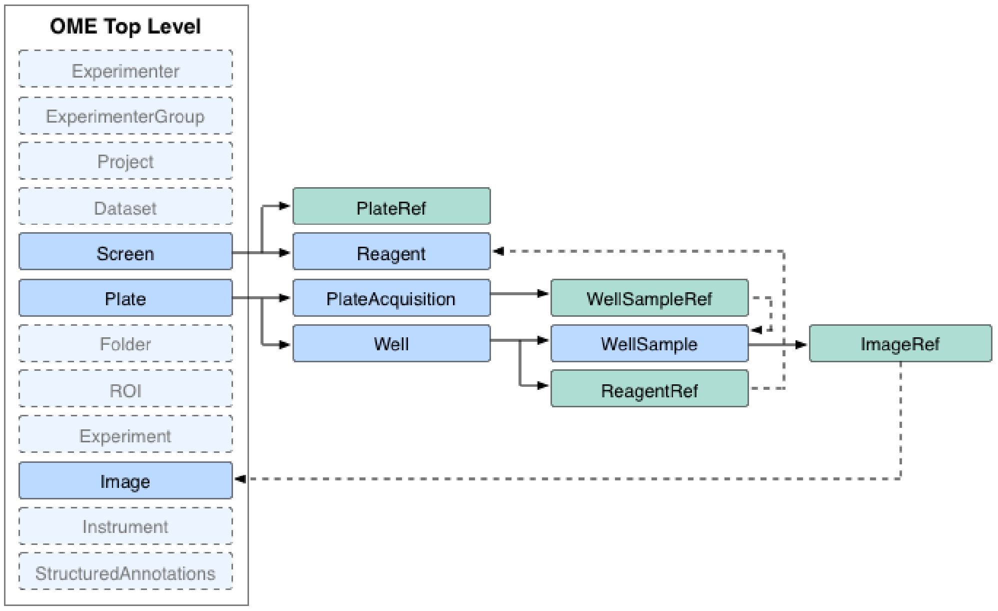


Fig. 14: SPW Schema structure

6.1.4 Structured Annotations

Structured annotations (or SAs), introduced in the *2008-09 schema*, are a general way to extend OME-XML with additional structured information. They can express a variety of data types and linkages, and serve as a replacement for the Custom Attributes and Semantic Type Definitions functionality of prior schemas.

For more information on SAs from an OMERO-centric perspective, see the OMERO page on structured annotations.

Map Annotations, storing 'key-value pairs', are a type of Structured Annotation which were introduced in the *Changes for January 2015*. Further information is available in the OMERO developer documentation on Key-value pairs and a sample OME-XML file is also available.

The structure of the SA used in the schema is shown below, along with all the possible attachment points in the model.

6.1.5 ROI model

ROIs are a top level object within the OME node of the model. This means that they can be referred to from more than one point within the model:

- They can be referenced by an Image node.
- They can be referenced by a MicrobeamManipulation node.
- They can be standalone - we interpret this as meaning they are a template that could be applied by the user to images.

Note: A fourth case is possible but not recommended. It is possible for one ROI to be referenced by both an Image node and a MicrobeamManipulation node. The problem with this is the ROI for the MicrobeamManipulation should be fixed, but the ROI on an Image is something that should be editable by a user. In this case, the user can inadvertently change the ROI set during the MicrobeamManipulation without realizing it.

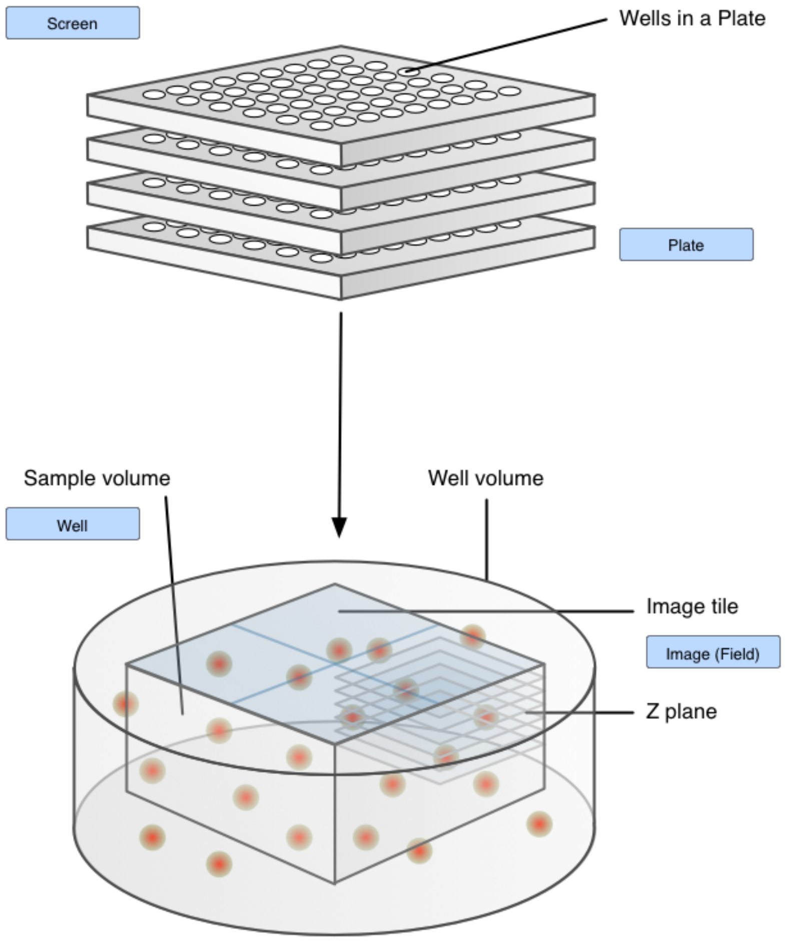


Fig. 15: Inter-dependencies of objects in the sample file structure

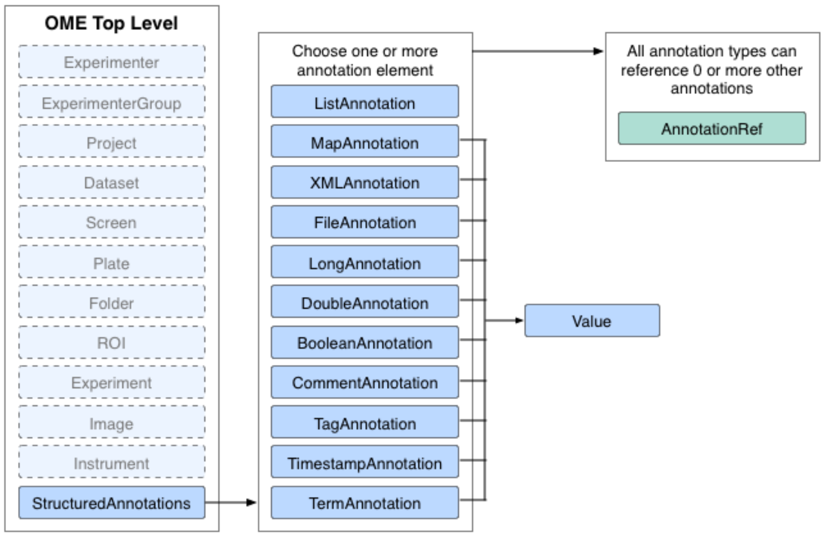


Fig. 16: The StructuredAnnotation branch of the OME Model

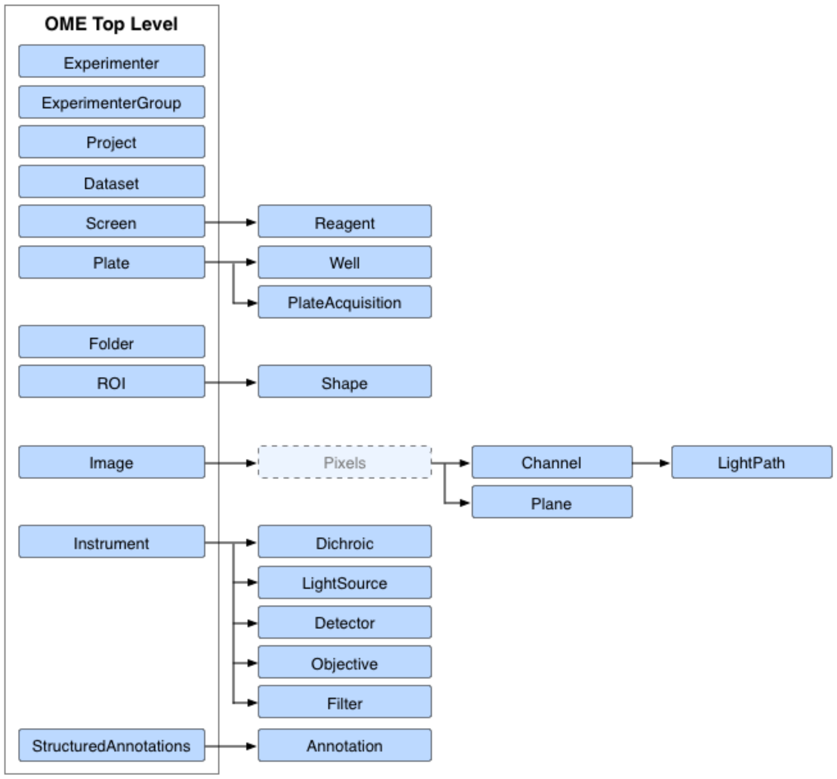


Fig. 17: All points in the OME Model that can be Annotated

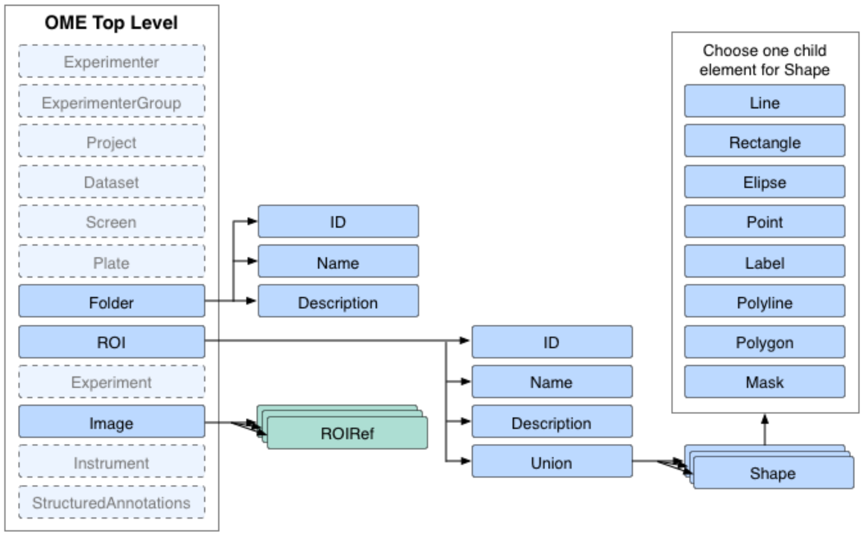


Fig. 18: ROI Model Overview

ROI attributes and simple children

The ROI node has some basic properties attached to it:

- An ID used to reference it from the Image and MicrobeamManipulation nodes.
- A short name for the ROI used in the user interface (optional).
- A longer description for the ROI used in the user interface (optional).
- An annotation reference linking an annotation to this ROI (optional).

ROI complex children

The ROI node has a choice of ONE child operation node. At the moment, the only choice is Union, meaning it is composed of the union of all its child shapes. It is implemented as a choice so we have the option available of adding other composition methods in the future. There are currently no plans in place for this however.

Shape types

The shape types define the geometry and appearance of the ROI. Each shape is a 2D object that exists within a single Z plane of an Image. (This will change with a future version of the schema).

Shape attributes and simple children

The shape abstract type has four groups of information which are attributes or simple children.

- General
  - A short name for the Shape used in the user interface (optional).
- Links to Planes
  - TheZ - the z-section this Shape is on (optional, if not specified then all the z-sections).
  - TheT - the timepoint this Shape is at (optional, if not specified then all the timepoints).
  - TheC - the channel this Shape is on (optional, if not specified then all the channels).
- Shape Display Options
  - FillColor - the color of the fill - encoded as RGBA (optional).
  - FillRule - which parts of the Shape to fill (optional).
  - StrokeColor - the color of the stroke (optional).
  - StrokeWidth - the width of the stroke in pixels (optional). This also has an optional length unit, StrokeWidthUnit.
  - StrokeDashArray - e.g. “none”, “10 20 30 10” (optional).
  - Text - a text label that can optionally be displayed on the Shape (optional).
  - FontFamily - the font family used to draw the text (optional).
  - FontSize - the size of the font in points (optional). This also has an optional length unit, FontSizeUnit.
  - FontStyle - the style applied to the text (optional).
- Geometry Adjustment
  - Transform - a transformation matrix represented by 6 values (optional).

Note: More information on transforms is available in this blog post.

Shape concrete implementations

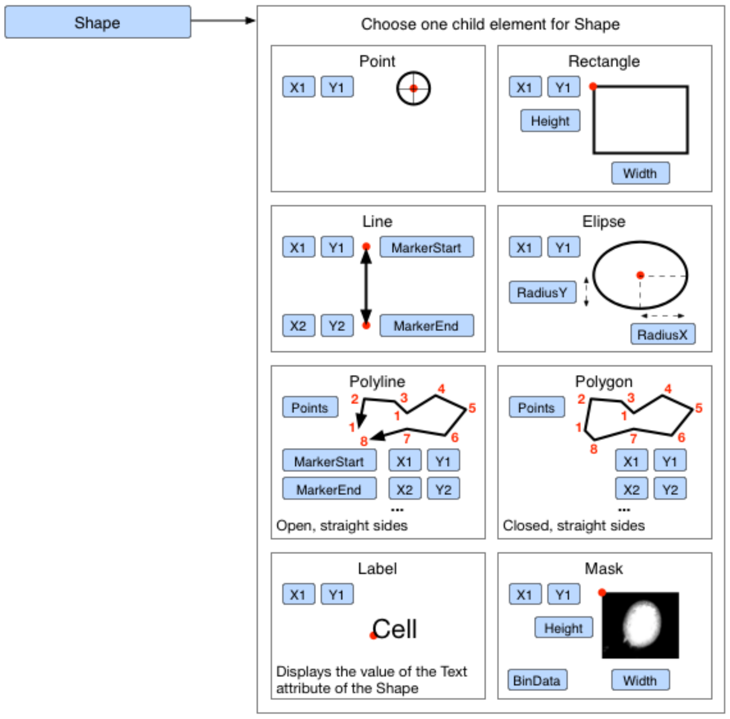


Fig. 19: Shape Implementations

The Shape abstract type has eight geometry implementations. At the moment the choice is:

- Ellipse - specified by a centre point, a radius in the X-axis, and a radius in the Y-axis.
- Label - specified by a start point for the baseline for the first character.
- Line - specified by two end points, a markerStart (an arrowhead marker applied to the start of the line) and markerEnd (an arrowhead marker applied to the end of the line) (optional).
- Mask - specified by an upper left corner and a BinData block.
- Point - a simple x, y position.
- Polygon - specified by an array of coordinates that are connected by straight lines that are closed, a markerStart (an arrowhead marker applied to the start of the line) and markerEnd (an arrowhead marker applied to the end of the line) (optional).
- Polyline - specified by an array of coordinates that are connected by straight lines, a markerStart (an arrowhead marker applied to the start of the line) and markerEnd (an arrowhead marker applied to the end of the line) (optional).
- Rectangle - specified by an upper left corner and a width and height.

3D ROI

The current method of defining a ROI in three dimensions is as a Union of Shape objects, each of which defines the geometry where that 3D ROI would cut the 2D Plane the Shape is attached to.

6.1.6 6D, 7D and 8D storage

Overview

This outlines an interim storage solution for storing 6D, 7D and 8D data in our 5D structure using Z, T and C.

This has been produced as part of our transition to N-dimensional support. It will take a large amount of work throughout the OME specifications and OME software to move to a fully N-dimensional approach to data storage. Given the scale and impact of the change, this will take some time to schedule and complete.

This proposal is designed to allow people wishing to write data with 6, 7 or 8 dimensions into our model today, in a way that can be upgraded in the future once N-dimensional support is available.

The key addition is a new XML Annotation to the Image element.

This was originally added to the pixels element but in use Image has proved to be a better location. We recommend Image is used from now on, but our code will also work with attachment to Pixels for backward compatibility.

This annotation will use namespace

```
"openmicroscopy.org/omero/dimension/modulo"
```

and store information on the additional dimensions embedded in the Z, T or C data. This annotation is optional and, if absent the model works as it currently does in the 5D case. Also, if an application does not understand the Modulo addition, then it can treat the data as 5D, though the displayed results would need some interpretation.

```
<SA:StructuredAnnotations>
    <SA:XMLAnnotation ID="Annotation:3" Namespace="openmicroscopy.org/omero/dimension/
↪modulo">
        <SA:Value>
            <Modulo namespace="http://www.openmicroscopy.org/Schemas/Additions/2011-09">
                <ModuloAlongZ Type="angle" Unit="degree">
                    <Label>45</Label>
                    <Label>90</Label>
                </ModuloAlongZ>
                <ModuloAlongT Type="lifetime" TypeDescription="TCSPC" Start="0" Step="2"␣
↪End="128"/>
                <ModuloAlongC Type="phase" Start="0" Step="1" End="255"/>
            </Modulo>
        </SA:Value>
    </SA:XMLAnnotation>
</SA:StructuredAnnotations>
```

The three dimensions ModuloAlongZ, ModuloAlongT and ModuloAlongC each can be specified in two ways:

- either using a specific number of labels (see ModuloAlongZ example above)
- or using a Start, Step and End (see ModuloAlongT or ModuloAlongC example above).

The attribute Type is **Required** and is drawn from the following enumeration:

- angle
- phase
- tile
- lifetime
- lambda
- other

The OMERO clients cannot display dimensions of type `other` but the OMERO server can store this data.

The attribute `TypeDescription` is optional. It is a simple text description of the Type used to indicate to an application or user how the data should be interpreted.

The attribute `Unit` is optional. It is a simple text description.

If you are going to specify `Label` elements inside the dimension then you do not need any other attributes.

If you are not using `Label` elements then you **MUST** specify the `Start` and `End` of the range the dimension covers. The dimension will be assumed to have values covering `Start` to `End` **INCLUSIVE**. The values will be assumed to go up in steps of 1 unless the optional `Step` is set to another value.

e.g. a collection of planes taken at the same timepoint but from different angle might use:

```
<SA:StructuredAnnotations>
    <SA:XMLAnnotation ID="Annotation:3" Namespace="openmicroscopy.org/omero/dimension/
↪modulo">
        <SA:Value>
            <Modulo namespace="http://www.openmicroscopy.org/Schemas/Additions/2011-09">
                <ModuloAlongZ Type="angle" Unit="degree">
                    <Label>45</Label>
                    <Label>90</Label>
                </ModuloAlongZ>
            </Modulo>
        </SA:Value>
    </SA:XMLAnnotation>
</SA:StructuredAnnotations>
```

The number of `Label` elements (if present) is used to devise the number of planes in the extra dimension. This is the equivalent of the value stored in `TheX`, `TheY`, `TheZ`, `TheT`, `TheC` in the 5D OME Model. If there are 2 `Label``s in ``ModuloAlongZ` then the dimension stores two planes for each Z. If there are no `Label` elements then `Start`, `Step` and `End` attributes are used to devise the number of planes in the extra dimension. So with Start 100, End 150 and Step 2 in ModuloAlongT then the dimension stores 26 planes for each T.

To find the true value of Z, T or C for a plane, you need to take the 5D value and divide it appropriately by the number of planes in the extra dimension.

e.g. if 3 extra planes are stored for each Z plane

```
| Number of Z Plane 5D view | True Z Plane | ModuleZ Plane |
|            0              |      0       |       0       |
|            1              |      0       |       1       |
|            2              |      0       |       2       |
|            3              |      1       |       0       |
|            4              |      1       |       1       |
|            5              |      1       |       2       |
|            6              |      2       |       0       |
```

(continues on next page)

(continued from previous page)

```
|            7              |       2      |       1       |
... and so on
```

e.g. if you have 2 extra Angle(A) planes stored for each Z plane and 3 extra Phase(P) for each T then

```
       Stored | Real
C1 , Z1 , T1  | C1 , Z1 , A1, T1 , P1
C2 , Z1 , T1  | C2 , Z1 , A1, T1 , P1
C1 , Z2 , T1  | C1 , Z1 , A2, T1 , P1
C2 , Z2 , T1  | C2 , Z1 , A2, T1 , P1
C1 , Z3 , T1  | C1 , Z2 , A1, T1 , P1
C2 , Z3 , T1  | C2 , Z2 , A1, T1 , P1
C1 , Z4 , T1  | C1 , Z2 , A2, T1 , P1
C2 , Z4 , T1  | C2 , Z2 , A2, T1 , P1
C1 , Z1 , T2  | C1 , Z1 , A1, T1 , P2
...
C2 , Z4 , T2  | C2 , Z2 , A2, T1 , P2
C1 , Z1 , T3  | C1 , Z1 , A1, T1 , P3
...
C2 , Z4 , T3  | C2 , Z2 , A2, T1 , P3
C1 , Z1 , T4  | C1 , Z1 , A1, T2 , P1
...
C2 , Z4 , T4  | C2 , Z2 , A2, T2 , P1
C1 , Z1 , T5  | C1 , Z1 , A1, T2 , P2
...
C2 , Z4 , T5  | C2 , Z2 , A2, T2 , P2
C1 , Z1 , T6  | C1 , Z1 , A1, T2 , P3
...
C1 , Z4 , T6  | C1 , Z2 , A2, T2 , P3
C2 , Z4 , T6  | C2 , Z2 , A2, T2 , P3
```

How to order the plane data

The order of the plane data is defined by the BinData or TiffData block, and is interleaved as it would be for the 5D view of the Z plane i.e. it is governed by the value of DimensionOrder on the Pixels element.

How to represent tiles

The Plane element stores the location of each tile.

e.g. Define four tiles 160 by 220 laid out as

```
---------
| A | B |
---------
| C | D |
---------


A   <Plane TheC="0" TheT="8"  TheZ="0" PositionX="0"   PositionY="0"   PositionZ="0.1"/>
B   <Plane TheC="0" TheT="9"  TheZ="0" PositionX="160" PositionY="0"   PositionZ="0.1"/>
```

(continues on next page)

(continued from previous page)

```
C   <Plane TheC="0" TheT="10" TheZ="0" PositionX="0"   PositionY="220" PositionZ="0.1"/>
D   <Plane TheC="0" TheT="11" TheZ="0" PositionX="160" PositionY="220" PositionZ="0.1"/>
```

Specifying the position of each plane allows tiles to either form a mosaic as above or to overlap e.g.

```
---------
| A | B |
---[E]---
| C | D |
---------

E   <Plane TheC="0" TheT="12" TheZ="0" PositionX="80" PositionY="110" PositionZ="0.1"/>
```

Sample files

Sample files are available in the *Modulo datasets* section of the OME-TIFF sample data.

6.1.7 Legacy use-case support

SPIM initial support

Note: This approach is still valid but has been **superseded** by the *6D, 7D and 8D storage*.

This proposal was put together after talks with various groups from the Single (or Selective) Plane Illumination Microscopy (also known as Light Sheet Microscopy) community, about how they can store data from this new and expanding field in the OME data formats.

Rather than a new full schema release, it was a common method of storing SPIM data in an existing schema. This was to allow users to start writing data from their SPIM systems right away in a way that could be upgraded to a full schema with SPIM support when it was released.

The solution below allowed the creation of valid OME-XML or OME-TIFF files that hold the extra SPIM data. We proposed to write an XSLT (Transform) that would upgrade these files to the schema version with the SPIM data moved to the correct location. What is described here is a file **only** solution. When the data is imported into an OMERO server, all the data will be retained as annotations but it cannot be fitted into the existing database model. This can result in some lost linkages between the data. This is be highlighted below where applicable.

Location in Model

SPIM data was placed in 6 key areas in the model:

1. Tiling
2. Illumination objectives and Cylindrical Lenses
3. Additional values for Objectives
4. Additional Stage Positions
5. SPIM angle
6. Custom system values

Tiling

Where the SPIM data has tiled images, the existing tiling method was used (see *Tiled images*).

This solution used StageLabel and the Position values on Plane to record the location and arrangement of the individual tiles.

```
<Image ID="Image:0" Name="Spim Sample Tile 1 Angle 1">
    <StageLabel Name="(1,1) of 1x2" X="1.00" Y="1.00"/>
    <Pixels>
        <Plane TheZ="1" TheT="1" TheC="1" PositionX="1.03" PositionY="0.98" PositionZ="1.
↪02"/>
...
</Image>
<Image ID="Image:1" Name="Spim Sample Tile 2 Angle 1">
    <StageLabel Name="(1,2) of 1x2" X="1.00" Y="2.00"/>
    <Pixels>
        <Plane TheZ="1" TheT="1" TheC="1" PositionX="1.03" PositionY="50.05" PositionZ=
↪"1.02"/>
...
</Image>
```

Illumination objectives and Cylindrical Lenses

The additional objective and other elements in the excitation light path were stored as additional Objectives within the Instrument block. This was already designed to support multiple Objectives.

```
<Objective ID="Objective:1" Immersion="Oil" LensNA="0.95" NominalMagnification="20"/>
<Objective ID="Objective:TubeLens:1" Model="OME - Tube lens 120 mm Mk1"/>
<Objective ID="Objective:Illumation:1" Model="Objective Mk2" LensNA="0.5"␣
↪WorkingDistance="19800"/>
```

Additional values for Objectives

As the function of Objective had been expanded to encompass the additional optical elements, some new attributes were required. These could not be attached directly to the Objective, so were instead attached as Annotations onto the Images that use the objectives. This was not ideal but was required due to restrictions on which objects can have annotations.

The definition of a single XmlAnnotation per Instrument was suggested, with each Image using that Instrument making use of an AnnotationRef to point at this Annotation.

The namespace of the XmlAnnotation had to be:

```
Namespace="ome-xml.org:additions:post2010-06:objective"
```

The Annotation would contain a single node that would define multiple Objectives, each with an ID matching one of the Objectives in the Instrument, and any additional attributes needed from the following list:

- FocalLength
- SlitAperture
- LightSheetWidth

Warning: Limitation - As the IDs of any Objective in the Instrument would not survive import into OMERO, these values, while imported as an annotation, would not be correctly linked.

```
<Image>
...
    <SA:AnnotationRef ID="Annotation:ObjectiveAdditions:1"/>
</Image>

<SA:XMLAnnotation ID="Annotation:ObjectiveAdditions:1" Namespace="ome-xml.
↪org:additions:post2010-06:objective">
    <SA:Description>Extra attribute values for the objective objects.</SA:Description>
    <SA:Value>
        <OME-Extra:ObjectiveAttributes xmlns:OME-Extra="http://www.openmicroscopy.org/
↪Schemas/Additions/2010-10">
            <ObjectiveAdditions ID="Objective:TubeLens:1" FocalLength="120" SlitAperture=
↪"6" LightSheetWidth="0.8"/>
            <ObjectiveAdditions ID="Objective:Illumation:1" FocalLength="25"/>
        </OME-Extra:ObjectiveAttributes>
    </SA:Value>
</SA:XMLAnnotation>
```

Additional Stage Positions

As there were several stages (Sample, Excitation Objective, Cylinder lens) that could be moved on a plane by plane basis, it was necessary to store these extra positions. It was proposed to use a collection of StagePosition objects, stored inside an XmlAnnotation linked to Image using an AnnotationRef.

The namespace of the XmlAnnotation **must** be:

```
Namespace="ome-xml.org:additions:post2010-06:spim:positions"
```

The Annotation would contain a single node that would define multiple StagePosition objects. Each of these objects would contain:

- TheZ, TheT, TheC to define the exact plane it was for
- Name, used to identify which stage the position was e.g. "objective_stage", "excitation_stage", "cylinder_stage"
- The PositionX, PositionY, PositionZ location of the stage. Not all of these position values needed to be present.

SPIM angle

The images collected for SPIM are recorded from a number of discrete angles. The angle for each image must be recorded. It was proposed that all the planes for the Z, C, T of each stack of images were stored in their correct location inside a single Image object. Multiple Image objects would then be used, each one repressing a single angle. The order and angle of these separate Images would then be defined by a single XmlAnnotation which each of the Images would reference using an AnnotationRef.

The namespace of the XmlAnnotation **must** be:

```
Namespace="ome-xml.org:additions:post2010-06:spim:set"
```

The Annotation would contain a single `<OME-Extra:SpimSet>` node that would define multiple SpimImage objects, each with an ID matching one of the Image objects that referenced the Annotation and an additional attribute:

- Angle

Warning: Limitation - although the IDs of any Image in the file would not survive import into OMERO, the linkages between these images could be maintained by adding the SpimSet annotation to each Image, as in the example below. Once imported, the single XML annotation would be linked to the imported images.

```
<Image ID="Image:0">
...
    <SA:AnnotationRef ID="Annotation:SpimSet:1"/>
</Image>
<Image ID="Image:1">
...
    <SA:AnnotationRef ID="Annotation:SpimSet:1"/>
</Image>

<SA:XMLAnnotation ID="Annotation:SpimSet:1" Namespace="ome-xml.org:additions:post2010-
↪06:spim:set">
    <SA:Value>
        <OME-Extra:SpimSet xmlns:OME-Extra="http://www.openmicroscopy.org/Schemas/
↪Additions/2010-10">
            <SpimImage ID="Image:0" Angle="0"/>
            <SpimImage ID="Image:1" Angle="45"/>
        </OME-Extra:SpimSet>
    </SA:Value>
</SA:XMLAnnotation>
```

Custom system values

At the time this was proposed, most SPIM systems were prototypes using custom software to drive them. If these systems required additional values to be stored in the file that do not fit within the above expansion of the OME model, then they should use their own Annotation, probably an XML annotation and define their own annotation namespace to use. This would be read and imported (and upgraded) into Bio-Formats and OMERO like any other annotation.

Sample

A hand written sample file is available that illustrates how the data can be structured. It is an OME-XML file but the broad structure of the metadata is the same for an OME-TIFF.

2010-06/spim.ome.xml

This file defines an instrument with multiple light sources and objectives.

```
<Instrument ID="Instrument:SpimSampleMicroscope1">
...
<LightSource ID="LightSource:1" Model="Laser Mk1" Manufacturer="OME-Sample"
            SerialNumber="LASER-1">
...
```

(continues on next page)

(continued from previous page)

```
<Objective ID="Objective:TubeLens:1" Manufacturer="OME-Sample"
            Model="OME - Tube lens 120 mm Mk1"/>
```

There are four image nodes each representing one SPIM angle.

```
<Image ID="Image:0" Name="Spim Sample Tile 1 Angle 1">
...
<Image ID="Image:1" Name="Spim Sample Tile 2 Angle 1">
...
<Image ID="Image:2" Name="Spim Sample Tile 1 Angle 2">
...
<Image ID="Image:3" Name="Spim Sample Tile 2 Angle 2">
```

They are connected together using a SpimSet annotation.

```
<SA:XMLAnnotation ID="Annotation:SpimSet:1" Namespace="ome-xml.org:additions:post2010-
↪06:spim:set">
```

Each image contains two channels, one named ‘Autoflouresence’, and one named ‘Green-OME’.

```
<Channel ID="Channel:0.0" Fluor="Autofluorescence" Color="-1"/>
<Channel ID="Channel:0.1" Fluor="Green-OME" Color="16711935"/>
```

Each image has 2 timepoints and two z-sections and is a small 6 pixel x 4 pixel image. This allows the BinData to be very small to allow you to focus on the file structure.

```
<Pixels DimensionOrder="XYCZT" ID="Pixels:0:0" PhysicalSizeX="10000.0"
    PhysicalSizeY="10000.0" PhysicalSizeZ="0.0" Type="uint8" SizeC="2" SizeT="2" SizeX="6
↪"
    SizeY="4" SizeZ="2">
...
<Bin:BinData BigEndian="false" Length="32"
    >/wCrzur//wB5oMPi/wBIbJO3AP8ePGCF</Bin:BinData>
```

There are extra annotations for the stage positions for each image and the extra objective attributes for each objective.

```
<SA:XMLAnnotation ID="Annotation:ExtraStageLabel:1:0"
...
<SA:XMLAnnotation ID="Annotation:ExtraStageLabel:1:1"
...
<SA:XMLAnnotation ID="Annotation:ExtraStageLabel:1:2"
...
<SA:XMLAnnotation ID="Annotation:ExtraStageLabel:1:3"
...
<SA:XMLAnnotation ID="Annotation:ObjectiveAdditions:1"
```

The file is valid and can be opened with OMERO.importer and Bio-Formats.

Tiled images

Storing tiled images in an OME file.

Note: This approach is still valid but has been **superseded** by the *6D, 7D and 8D storage* solution.

When you are looking at a sample that is larger than the available field of the microscope, one approach is to acquire a number of images, moving across the sample. These individual image pieces can then be joined together at a later date to form an image of the entire sample.

There are two structures in the model that support the joining process:

- **StageLabel** - part of Image
- **Position** - part of Plane

The mainstay of the OME format is the 5-dimensional pixel array. In order to maintain the integrity of this structure, the Tiling process has to be external to the Image structure. This means that you represent the Tiled image as a collection of Image elements. It is up to the software loading the OME file to interpret these individual Image elements as Tiles.

Each Image element can have a StageLabel. This is a named point in the co-ordinates of the microscope reference frame. The StageLabel has a Name attribute, that can store a human-readable description of the point, as well as X, Y, and Z values for the co-ordinates. Each co-ordinate is optional so an (X,Y) position or (X,Z) position can be recorded if required. This is probably best thought of as the target location for the individual tile.

During the acquisition process, the field is moved to capture each set of pixels making up a plane of an image of the tiled image. The exact location of the plane is recorded in the PositionX, PositionY, PositionZ on Plane. The Position values in Plane should correspond to the values in StageLabel but, depending on the instrument and capture method, there may be small discrepancies.

Note: Some systems in the past have used the Time value as a place to store the tile number of an image. This not the correct approach in the OME model.

A sample file showing the structure is:

```
<?xml version="1.0" encoding="UTF-8"?>
<OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2012-06"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2012-06
                        http://www.openmicroscopy.org/Schemas/OME/2012-06/ome.xsd">
    <!-- This is a dummy file - it has no real data -->
    <!-- File to represent a tiled image aligned in a 2 by 2 grid -->
    <!-- as a collection of 4 images a, b, c, and d.  -->
    <!-- 0 | 1 | 2 | -->
    <!-- - | - | - | -->
    <!-- 1 | a | b | -->
    <!-- 2 | c | d | -->
    <!-- - | - | - | -->
    <!-- First image - this will be the tile at 1,1 on the grid -->
    <Image ID="Image:a" Name="2x2 Image">
        <AcquisitionDate>2008-03-01T18:13:51.0Z</AcquisitionDate>
        <StageLabel Name="(1,1) of 2x2" X="1.00" Y="1.00"/>
        <!-- X and Y (and Z if present) are the target location -->
        <Pixels ID="Pixels:a" DimensionOrder="XYZCT" Type="uint8"
```

(continues on next page)

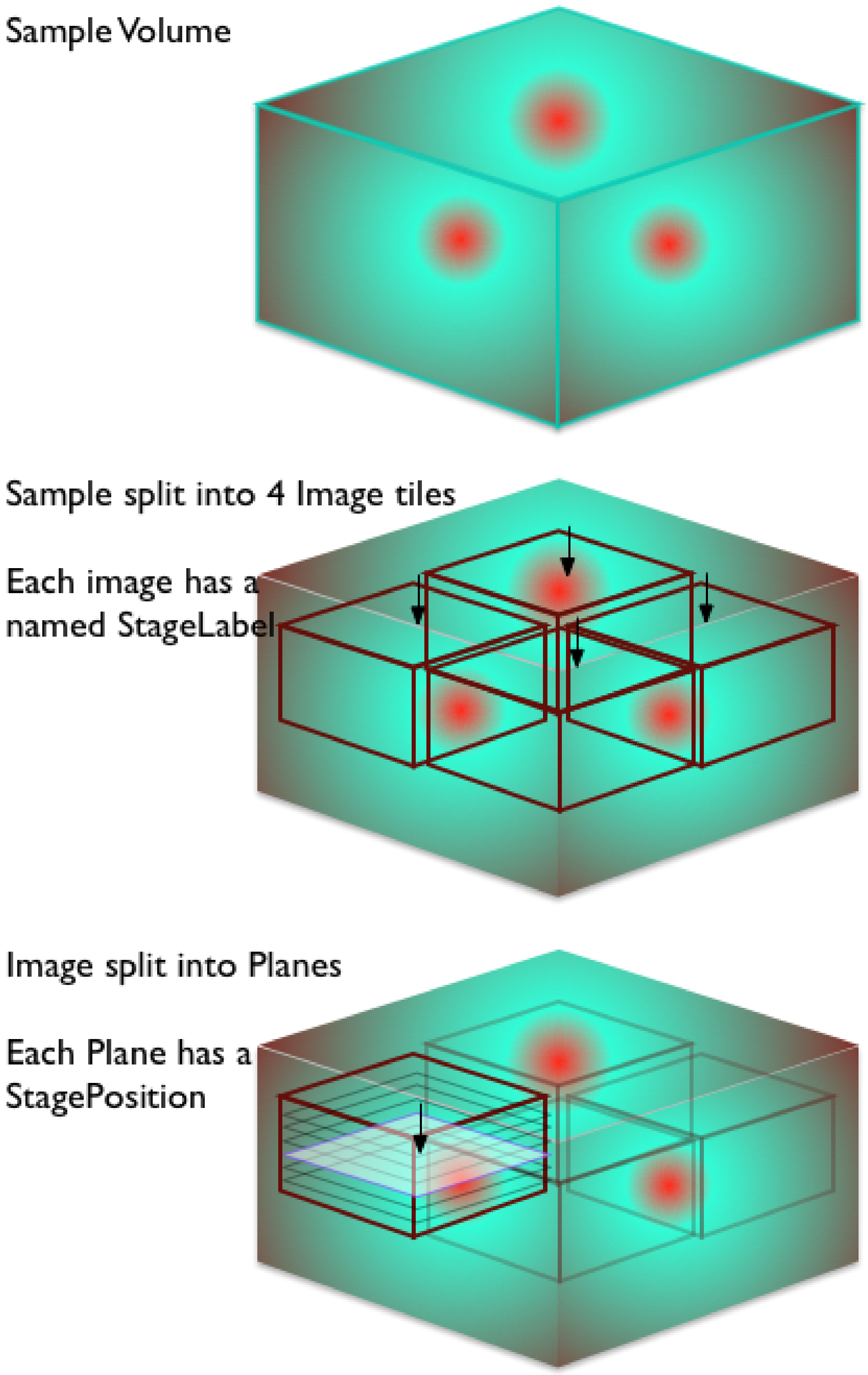

Sample Volume
Sample split into 4 Image tiles
Each image has a named StageLabel
Image split into Planes
Each Plane has a StagePosition

Fig. 20: Tile Samples Image

(continued from previous page)

```
            SizeX="128" SizeY="128" SizeZ="8" SizeC="1" SizeT="3">
            <BinData xmlns="http://www.openmicroscopy.org/Schemas/BinaryFile/2012-06"
                Length="10" BigEndian="true">
                <!-- ... -->
            </BinData>
            <Plane TheZ="1" TheT="1" TheC="1" DeltaT="0.01" ExposureTime="0.004"
                PositionX="1.03" PositionY="0.98" PositionZ="1.02" >
                <!-- X, Y and Z are the actual location when the plane was acquired-->
            </Plane>
            <Plane TheZ="2" TheT="1" TheC="1" DeltaT="0.02" ExposureTime="0.004"
                PositionX="1.03" PositionY="0.98" PositionZ="1.23">
            </Plane>
            <!-- ... and so on for the other Z sections -->
            <!-- ... and then for the other Time points -->
        </Pixels>
    </Image>

    <!-- Second image - this will be the tile at 1,2 on the grid -->
    <Image ID="Image:b" Name="2x2 Image">
        <AcquisitionDate>2008-03-01T18:13:51.0Z</AcquisitionDate>
        <StageLabel Name="(1,2) of 2x2" X="1.00" Y="2.00"/>
        <!-- X and Y (and Z if present) are the target location -->
        <Pixels ID="Pixels:b" DimensionOrder="XYZCT" Type="uint8"
            SizeX="128" SizeY="128" SizeZ="8" SizeC="1" SizeT="3">
            <BinData xmlns="http://www.openmicroscopy.org/Schemas/BinaryFile/2012-06"
                Length="10" BigEndian="true">
                <!-- ... -->
            </BinData>
            <Plane TheZ="1" TheT="1" TheC="1" DeltaT="0.31" ExposureTime="0.004"
                PositionX="1.02" PositionY="2.01" PositionZ="1.01">
                <!-- X, Y and Z are the actual location when the plane was acquired-->
            </Plane>
            <Plane TheZ="2" TheT="1" TheC="1" DeltaT="0.32" ExposureTime="0.004"
                PositionX="1.02" PositionY="2.01" PositionZ="1.24">
            </Plane>
            <!-- ... and so on for the other Z sections -->
            <!-- ... and then for the other Time points -->
        </Pixels>
    </Image>

    <!-- Third image - this will be the tile at 2,1 on the grid -->
    <!-- ... -->

    <!-- Fourth image - this will be the tile at 2,2 on the grid -->
    <!-- ... -->
</OME>
```

An alternative valid form would have `TiffData` blocks instead of the `BinData` blocks. This would be used in the header of an OME-TIFF file.

CHAPTER

SEVEN

DATA MODEL HISTORY

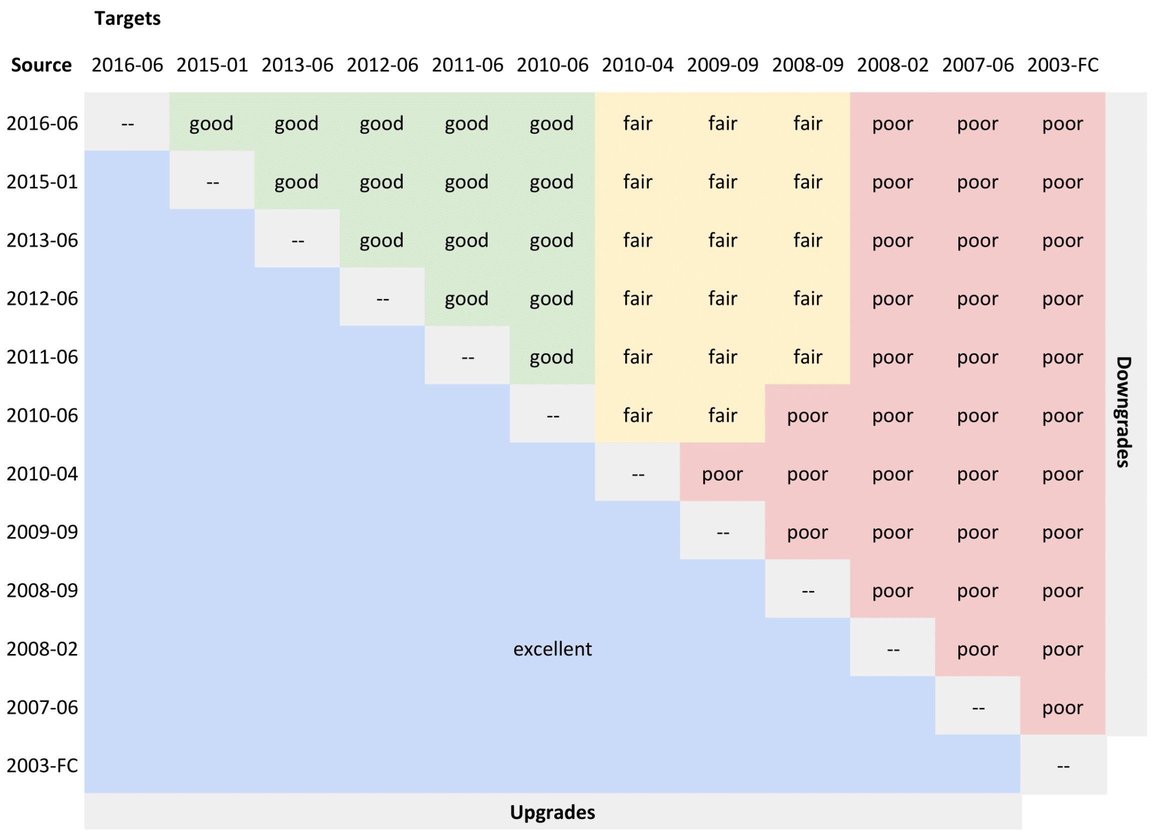
7.1 Schema version information

7.1.1 Transformations

Available transformations

Available transforms	Direction	Status
2003-FC-to-2007-06.xsl	upgrade	*excellent*
2003-FC-to-2008-09.xsl	upgrade	*excellent*
2007-06-to-2008-02.xsl	upgrade	*excellent*
2007-06-to-2008-09.xsl	upgrade	*excellent*
2008-02-to-2008-09.xsl	upgrade	*excellent*
2009-09-to-2010-04.xsl	upgrade	*excellent*
2010-04-to-2010-06.xsl	upgrade	*excellent*
2010-04-to-2010-06.xsl	upgrade	*excellent*
2010-06-to-2011-06.xsl	upgrade	*excellent*
2011-06-to-2012-06.xsl	upgrade	*excellent*
2012-06-to-2013-06.xsl	upgrade	*excellent*
2013-06-to-2015-01.xsl	upgrade	*excellent*
2015-01-to-2016-06.xsl	upgrade	*excellent*
2010-06-to-2003-FC.xsl	downgrade	*poor*
2010-06-to-2008-02.xsl	downgrade	*fair*
2011-06-to-2010-06.xsl	downgrade	*good*
2012-06-to-2011-06.xsl	downgrade	*good*
2013-06-to-2012-06.xsl	downgrade	*good*
2015-01-to-2013-06.xsl	downgrade	*good*
2016-06-to-2015-01.xsl	downgrade	*good*

Quality of transformations



Targets

Source	2016-06	2015-01	2013-06	2012-06	2011-06	2010-06	2010-04	2009-09	2008-09	2008-02	2007-06	2003-FC
2016-06	--	good	good	good	good	good	fair	fair	fair	poor	poor	poor
2015-01		--	good	good	good	good	fair	fair	fair	poor	poor	poor
2013-06			--	good	good	good	fair	fair	fair	poor	poor	poor
2012-06				--	good	good	fair	fair	fair	poor	poor	poor
2011-06					--	good	fair	fair	fair	poor	poor	poor
2010-06						--	fair	fair	poor	poor	poor	poor
2010-04							--	poor	poor	poor	poor	poor
2009-09								--	poor	poor	poor	poor
2008-09									--	poor	poor	poor
2008-02						excellent				--	poor	poor
2007-06											--	poor
2003-FC												--

Upgrades

Downgrades

Key to quality

poor
: the bare minimum of metadata is preserved to allow image display, all other metadata is lost

fair
: a portion of the metadata is preserved, at least enough to display the image and some other data, it will be far from complete however

good
: most information is preserved, it may be possible to do a better job but could be difficult for technical reasons or require custom code not just a transform

excellent
: as much information as possible is preserved, some values can still be lost if there are completely incompatible with the new schema

7.1.2 Changes for June 2016 release 2

List of the key changes made for the minor release updating the June 2016 ome-xml data model. These changes were introduced with the release of Bio-Formats 5.2.3 in October 2016.

The new minor release of the schema has the same namespace (2016-06) and a new version number. As a minor release, any file that validated correctly using the last major release will also validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers to understand the changes.

- The version number of the `ome.xsd` schema is now 2.

Overview of changes

Channel

Added the following as valid values for `AcquisitionMode`:

- `BrightField`
- `SweptFieldConfocal`
- `SPIM`

Also added parsing for Laser Scan Confocal and Swept Field Confocal.

7.1.3 Changes for June 2016

The list of the key changes for the June 2016 major release of the OME-XML data model. This schema release will tie in with the Bio-Formats 5.2 release.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

This schema uses the new namespace:

```
http://www.openmicroscopy.org/Schemas/OME/2016-06/
```

All the schema files have been unified into a single `ome.xsd` file located at this namespace. The version number of this schema file is 1.

Overview of changes

- This release will be implemented in Bio-Formats 5.2 and OMERO 5.3.
- This release introduces the concept of Folders, a new model object which may contain Images, ROIs and other Folders, and which has a strict tree hierarchy.

Details

- The model element `Folder` was added. A Folder specifies a possibly heterogeneous collection of data and may contain other Folders, Images and ROIs. Data may be in multiple Folders but a Folder may not be in more than one other Folder, giving a strict tree hierarchy. The primary driver for this addition is to allow the organization of ROI elements into hierarchical structures (see this Folders blog post for further discussion).
- The ROI properties `ROI.Namespace`, `Shape.Linecap` and `Shape.Visible` have been dropped. This simplifies graphical aspects in the data model in favor of more generic enumerated values independent of the rendering framework, making it easier to implement across clients (see this Design issue for an example discussion).
- The `Marker` enumeration has been reduced to `Arrow` only, dropping `Circle` and `Square`. This affects the `Line.MarkerStart`, `Line.MarkerEnd`, `Polyline.MarkerStart` and `Polyline.MarkerEnd` attributes. As with the ROI property changes, these are also intended to simplify the graphical aspects of the data model.

The following are simplifications for code generation purposes and have no functional effects:

- The model elements `LightSource` and `Shape` have become abstract classes, meaning they are now complexTypes in the `.xsd`. This provides a more intuitive object-oriented design (e.g. Rectangle is a type of Shape rather than Shape containing a Rectangle) and simplifies the data representation (schema) as well as the code generation because two different ways to model inheritance and polymorphism have been changed to one.
- The `MapPairs` model element has been removed as an unnecessary container which is superseded by using the maps directly in the generated code.
- The `Map` model element has been made a complexType. This element was never used directly, only as a base, so the change allows for the removal of a redundant model object.
- xsd:appinfo has been extended to provide more detail for enumeration code generation, in particular for units.

Upgrading and Downgrading

The XSLT transforms between January 2015 and June 2016 versions are available here:

http://www.openmicroscopy.org/Schemas/Transforms/2015-01-to-2016-06.xsl

http://www.openmicroscopy.org/Schemas/Transforms/2016-06-to-2015-01.xsl

7.1.4 Changes for January 2015

The list of the key changes for the January 2015 major release of the OME-XML data model. This schema release will tie in with the Bio-Formats 5.1 release.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

The version number of all schema files is now 1, except ome.xsd which version number is 2.

This schema uses the new namespace:

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/2015-01/
```

For the OME schema

```
http://www.openmicroscopy.org/Schemas/OME/2015-01/
```

and the schema file is located at

```
http://www.openmicroscopy.org/Schemas/OME/2015-01/ome.xsd
```

Overview of changes

- This version introduces a major new system of specifying units for the values stored as lengths, times, pressures, angles, temperatures, electric potentials (voltages), powers and frequencies. For more information see *the OME system of units*.
- Many annotation points have been added and some removed. Objects in the model are now directly annotatable or have a 1 to 1 relationship with an object that is.

BinaryFile

- There are no significant changes to this component.

OME

- Expanded documentation for `AcquisitionDate` describing supported precision.
- Updated documentation for `NDFilter` to reflect usage
- **Added Annotation points (`AnnotationRef`) to:**

  ```
  Instrument
  Objective
  Detector
  Filter
  Dichroic
  LightPath
  LightSource
  ```

- **Added attributes to store:**

  ```
  Pixels:  PhysicalSizeXUnit
  Pixels:  PhysicalSizeYUnit
  Pixels:  PhysicalSizeZUnit
  Pixels:  TimeIncrementUnit
  Plane:  DeltaTUnit
  Plane:  ExposureTimeUnit
  Plane:  PositionXUnit
  Plane:  PositionYUnit
  Plane:  PositionZUnit
  Channel:  PinholeSizeUnit
  Channel:  ExcitationWavelengthUnit
  Channel:  EmissionWavelengthUnit
  StageLabel:  XUnit
  StageLabel:  YUnit
  ```

```
StageLabel:  YUnit
ImagingEnvironment:  TemperatureUnit
ImagingEnvironment:  AirPressureUnit
Objective:  WorkingDistanceUnit
Detector:  VoltageUnit
Filter:  CutInUnit
Filter:  CutOutUnit
Filter:  CutInToleranceUnit
Filter:  CutOutToleranceUnit
LightSource:  PowerUnit
Laser:  WavelengthUnit
Laser:  RepetitionRateUnit
LightSourceSettings:  WavelengthUnit
DetectorSettings:  VoltageUnit
DetectorSettings:  ReadOutRateUnit
```

- **Removed Annotation points from:**

```
Pixels:  AnnotationRef
```

- **Changed from int to float:**

```
Channel:  ExcitationWavelength
Channel:  EmissionWavelength
Filter:  CutIn
Filter:  CutOut
Filter:  CutInTolerance
Filter:  CutOutTolerance
Laser:  Wavelength
LightSourceSettings:  Wavelength
```

- Added a general `Map` to `ImagingEnvironment` to store key-value pairs
- Added a new core type `NonNegativeFloat`
- Added a new core element `Map` and associated complex type `MapPairs`. This in turn contains a collection of `M` elements that store a mapped value, each with its associated `K` key attribute.
- **Defined new enumerations with the permitted values for:**

```
UnitsLength
UnitsTime
UnitsPressure
UnitsAngle
UnitsTemperature
UnitsElectricPotential
UnitsPower
UnitsFrequency
```

- Added a new `GenericExcitationSource` to light source types. This uses a `Map` of key-value pairs to store metadata for a light source that cannot be expressed as one of the other types.

OMERO

`OMERO.xsd` is not included in this release.

ROI

- **Added Annotation points to:**
  `Shape: AnnotationRef`
- **Added attributes to store:**
  `ROI: StrokeWidthUnit` `ROI: FontSizeUnit`

SA

- Added a new `MapAnnotation` type. This makes use of the new `Map` element from `ome.xsd` to store a collection of key-value pairs.

SPW

- **Added attributes to store:**
  `WellOriginXUnit` `WellOriginYUnit` `PositionXUnit` `PositionYUnit`
- **Removed Annotation points from:**
  `WellSample: AnnotationRef`

Upgrading and Downgrading

The XSLT transforms between June 2013 and January 2015 versions are available here:

http://www.openmicroscopy.org/Schemas/Transforms/2013-06-to-2015-01.xsl

http://www.openmicroscopy.org/Schemas/Transforms/2015-01-to-2013-06.xsl

7.1.5 Changes for June 2013

The list of the key changes for the June 2013 major release of the ome-xml data model. This schema release will tie in with the Bio-Formats 5.0 release.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

The version number of all schema files is now 1.

This schema uses the new namespace:

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/2013-06/
```

For the OME schema

```
http://www.openmicroscopy.org/Schemas/OME/2013-06/
```

and the schema file is located at

```
http://www.openmicroscopy.org/Schemas/OME/2013-06/ome.xsd
```

Overview of changes

- This release is in step with the release of Bio-Formats 5.0 and OMERO 5.0 (FS).

BinaryFile

- There are no significant changes to this component.

OME

- Copyright information is stored in `RightsHolder` and `RightsHeld`.
- `SignificantBits`, `Interleaved`, and `BigEndian` attributes have been added to `Pixels`.
- `NominalMagnification` has been altered to allow non-integer values, e.g. 0.5.
- A new type has been added to``Filter`` called `Tuneable`.
- `Zoom` and `Integration` have been added to `DetectorSettings`.

OMERO

`OMERO.xsd` is not included in this release.

ROI

There are no significant changes to this component.

SA

A new optional `Annotator` attribute has been added to the `Annotation` complex type. This makes it available to all other annotation types. This attribute is of type `ExperimenterID` and can be used to record who created an annotation. It has not been added to the OMERO database as this information is already available through the permissions system.

SPW

There are no significant changes to this component.

Upgrading and Downgrading

The XSLT transforms between June 2012 and June 2013 versions are available here:

http://www.openmicroscopy.org/Schemas/Transforms/2012-06-to-2013-06.xsl

http://www.openmicroscopy.org/Schemas/Transforms/2013-06-to-2012-06.xsl

7.1.6 Changes for June 2012

The list of the key changes for the June 2012 major release of the ome-xml data model. This schema release will tie in with the Bio-Formats 4.4 release.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

The version number of all schema files is now 1.

This schema uses the new namespace:

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/2012-06/
```

For the OME schema

```
http://www.openmicroscopy.org/Schemas/OME/2012-06/
```

and the schema file is located at

```
http://www.openmicroscopy.org/Schemas/OME/2012-06/ome.xsd
```

Overview of changes

- This release is in step with release of Bio-Formats 4.4.
- It is preparation for the Major release of OMERO **after** 4.4.
- It includes work to allow full code generation of OME Model API direct from the XSD files.
- It also includes almost full synchronization of the OME Model with the OMERO Database structure.
- The direction of references between Screen, Plate and Well, and between Project, Dataset and Image, have been reversed.
- The ROI model has been majorly reworked.
- `Pixels` has been **DEPRECATED**, but is still required at present.
- `OTF` and all associated objects have been **REMOVED**.

General changes

- **All** colors are now stored as new type `Color`.
- The default value for all colors is now solid white (due to legacy color default it was totally transparent red).
- `appinfo` blocks for `xsd-fu` have been added to specify plurals.
- `appinfo` blocks for `xsd-fu` have been added to specify abstract objects.

BinaryFile

- The only change here is the general change adding `appinfo` blocks for `xsd-fu` to specify plurals.

OME

- `AcquiredDate` has been renamed to `AcquisitionDate` in `Image`.
- `ExperimentGroup` has been renamed to `Group`.
- `Group` is now annotatable.
- Multiple `Leader``s are now possible for each ``Group`.
- `Contact` has been removed from `Group` - the transform will convert any existing `Contact` to a `Leader`.
- The direction of the references between `Project`, `Dataset` and `Image` have been reversed for consistency with other linked objects.
- `DisplayName` has been removed from `Experimenter`.
- The `ImageRef` element has been moved from the `SPW.xsd` to the `ome.xsd` schema file.
- `Pixels` has been **DEPRECATED**, but is still required. The contents of `Pixels` will be moved up to `Image` in the next release - it is a legacy of when `Image` could have multiple `Pixels` sets and has been redundant since the 2009-09 schema fixed the number of `Pixels` sets in an `Image` at 1.
- `OTF` and all associated objects have been **REMOVED**.

OMERO

`OMERO.xsd` is not included in this release. It has been extensively reworked and expanded as part of the OME Model and OMERO Database synchronization but work on this will continue until the release of the associated OMERO version. The OME Model does not make use of `OMERO.xsd` so will be unaffected by the future release of this single file.

Note: Some differences must remain between OME Model and OMERO Database for database optimization reasons - these are handled by special cases in our code generation application `xsd-fu`.

ROI

- All objects defining colors now use the type 'OME:Color' from file:*ome.xsd*.
- `MarkerStart` and `MarkerEnd` have been moved down from `Shape` into `Line`, `Polyline`.
- A `Polygon` element (a new type of `Shape`) has been added.
- 'Closed' has been deleted from Polyline (a Closed Polyline is now transformed into a Polygon).
- The attribute `Name` has been deleted from `Shape`.
- `Description` has been deleted from `Shape`.
- `Visible` has been added to `Shape`.
- `Locked` has been added to `Shape`.
- The `Fill` attribute in `Shape` has been renamed to `FillColor`.
- The `Stroke` attribute in `Shape` has been renamed to `StrokeColor`.
- `Value` has been deleted from `Text`.
- The `Text` element has been renamed to `Label`.
- The `Label` attribute in `Shape` has been renamed to `Text`.
- `Transform` has been converted to the new type `AffineTransform`.

SA

Additional intermediate abstract annotations (`BasicAnnotation`, `NumericAnnotation`, `TextAnnotation`, `TypeAnnotation`) have been added to match OMERO annotation structure. The current annotations have been modified to now extend these. This allows the code generation to group annotations together by intermediate annotation type. The behavior of the current annotations has not changed, and the new intermediate abstract annotations are not used directly.

SPW

- `FieldIndex` has been added to `Plate` - this means the index of the WellSample displays as the default Field.
- The human readable identifier for the screening status has been renamed from 'Status' to 'Type'.
- The direction of the references between `Screen` and `Plate` has been reversed for consistency with other linked objects.
- The `ImageRef` element has been moved to the `ome.xsd` schema file.

Upgrading and Downgrading

The XSLT transforms between June 2011 and June 2012 versions are available here:

http://www.openmicroscopy.org/Schemas/Transforms/2011-06-to-2012-06.xsl

http://www.openmicroscopy.org/Schemas/Transforms/2012-06-to-2011-06.xsl

7.1.7 Changes for June 2011

The list of the key changes for the June 2011 major release of the ome-xml data model. This schema release will tie in with the OMERO 4.3 release.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

The version number of all schema files is now 1.

This schema uses the new namespace:

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/2011-06/
```

For the OME schema

```
http://www.openmicroscopy.org/Schemas/OME/2011-06/
```

and the schema file is located at

```
http://www.openmicroscopy.org/Schemas/OME/2011-06/ome.xsd
```

Overview of changes

- `Polyline` and `Path` have been **DEPRECATED**.
- ROI font enumerations have been updated.
- Metadata-Only Companion OME-XML and Binary-Only OME-TIFF files have been created.
- A `Creator` attribute has been added for OME-XML and OME-TIFF.
- The OTF (Optical Transfer Function) has been **DEPRECATED**.

ROI

- `Polyline` and `Path` have been **DEPRECATED** as they are due to be replaced in the next major release, to provide support for 3-dimensional objects.
- `FontFamily` has been changed so it matches the standard HTML/CSS values.
- `FontStyle` has been changed so it better matches standard HTML/CSS.

OME

- The OME node has been modified to allow creation of the Metadata-Only Companion OME-XML and Binary-Only OME-TIFF files.
- The OME node `Creator` attribute can now contain the name and version of the creating application.
- The physical size of a pixel is now restricted to a positive value.
- `MicrobeamManipulation` can now have a `Description`.
- OTF (Optical Transfer Function) has been **DEPRECATED** and is due for removal in the next major release.

7.1.8 Changes for June 2010

The list of the key changes for the June 2010 major release of the ome-xml data model. This schema release will tie in with the OMERO 4.2 release.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

The version number of all schema files is now 1.

This schema uses the new namespace:

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/2010-06/
```

For the OME schema

```
http://www.openmicroscopy.org/Schemas/OME/2010-06/
```

and the schema file is located at

```
http://www.openmicroscopy.org/Schemas/OME/2010-06/ome.xsd
```

General updates

This release represents a collection of general updates to solve problems.

- The release has been brought in step with release of OMERO 4.2.
- Integer types have been made more restrictive.
- A problem with `Well Sample Timepoint` has been fixed.
- `Width` and `Height` have been added to ROI `Mask`.
- Annotations have been reworked to add new types (‘Term’ and ‘Tag’) and allow cross-linking of all annotations.

ome.xsd

- Integer types have been made more restrictive.

ROI.xsd

- `Width` has been added to `Mask`.
- `Height` has been added to `Mask`.
- `AnnotationRef` has been removed from `Shape` (`Shape` is no longer directly annotatable, annotations will be moved up to the ROI level).

SA.xsd

- `CommentAnnotation` has been renamed as `StructuredAnnotations`.
- `TagAnnotation` has been added to `StructuredAnnotations`.
- `TermAnnotation` has been added to `StructuredAnnotations`.
- `Description` has been added to `Annotation`.
- The list of `AnnotationRef` has been added to `Annotation`.
- `Namespace` in `Annotation` is now optional.
- The list of `AnnotationRef` has been removed from `ListAnnotation` (it is now inherited from the base `Annotation`).
- `ROI:Shape` is no longer in the list of annotatable objects.

SPW.xsd

- `WellOriginX` in `Plate` has been documented.
- `WellOriginY` in `Plate` has been documented.
- `NamingConvention` in `Plate` has been documented.
- The type of `Timepoint` in `WellSample` is now an 'xsd:dateTime'.

7.1.9 Changes for April 2010

List of key changes for April 2010 major release of the ome-xml data model.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

The version number of all schema files is now 1.

This schema uses the new namespace

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/2010-04/
```

For the OME schema

```
http://www.openmicroscopy.org/Schemas/OME/2010-04/
```

and the schema file is located at

```
http://www.openmicroscopy.org/Schemas/OME/2010-04/ome.xsd
```

Overview of components changed

- schema cleanup
- units
- ROI changes
- SPW changes
- Filter changes

Schema cleanup

The following legacy objects have been removed:

- `AnalysisChain.xsd`
- `AnalysisModule.xsd`
- `CA.xsd`
- `CLI.xsd`
- `DataHistory.xsd`
- `MLI.xsd`
- `STD.xsd`

This required the removal of references to them from `ome.xsd`; the removal of `CustomAttributes` from OME, Image and Dataset nodes; and the removal of `SemanticTypeDefinitions` from the OME Node.

Units

- Units were defined for everything in
  - `ome.xsd`
  - `ROI.xsd`
  - `SPW.xsd`

Follow some examples:

```
Physical size of a pixel in microns[um]
The Z-section this plane is for. [units:none]
This is the name of the fluorophore used to produce this channel [plain text string]
```

ROI changes

- ROI now has:
  - an optional `Description` element
  - an optional `Name` attribute
  - an optional `Namespace`
- Shape now has:

  - an abstract marker in an appinfo
  - an optional `Name` attribute
  - `ChannelRef [0..\*]` removed
  - `TheC` added linked to `Channel [0..1]`
  - all the attributes previously on the dropped `ShapeDisplayOptions` e.g. `Fill`, `Stroke`, `FontFamily`, etc.
- The `Text` element has been renamed to `Label`.

SPW changes

- `Plate` now has:
  - `PlateAcquisition [0..\*]` added
  - an attribute `Rows`
  - an attribute `Columns`
  - a unique key `WellSampleIndex` added for the attribute `Index` in `SPW:Well/SPW:WellSample`
  - `DefaultSample` removed
- `ScreenAcquisition` has become `PlateAcquisition`.
- `PlateAcquisition` now has:
  - an optional `Description` element added
  - an optional `Name` attribute added
  - an optional `MaximumFieldCount` attribute added
- `Screen` now has `ScreenAcquisition` removed.
- `WellSample` now has `Index` added.

Filter changes

- Filter now has type 'Dichroic'.
- There is a new `LightPath` object (this provides much more flexibility than `SecondaryEmissionFilter` and `SecondaryExcitationFilter`).

Other

- There have been changes and additions made to `Detector`.
- The lack of LED attributes was explained in documentation.

7.1.10 Changes for September 2009

The list of the key changes for the September 2009 major release of the ome-xml data model.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

The version number of all schema files is now 1.

This schema uses the new namespace:

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/2009-09/
```

For the OME schema

```
http://www.openmicroscopy.org/Schemas/OME/2009-09/
```

and that the schema file will be located at

```
http://www.openmicroscopy.org/Schemas/OME/2009-09/ome.xsd
```

Overview of changes

Additional Schema file ROI.xsd

This provides support for the new version of the ROI objects. These were moved from the OME namespace to there own namespace and the objects updated to be simpler and more consistent. Existing ROI's can be updated to the new structure.

- Some support for ROIs in `OMERO.xsd` has moved to be part of the full schema in `ROI.xsd`.

MicrobeamManipulation

- `MicrobeamManipulation` has been moved from `Image` to `Experiment`.

At present `MicrobeamManipulation` is stored directly under an `Image` with a link to an `Experiment`. The structure this produces is too flexible and allow loops of references to be created. It is proposed that we move all the `MicrobeamManipulation` objects to be located under `Experiment` and replace them in each `Image` with zero or more `MicrobeamManipulationRef` objects. This also allows one `MicrobeamManipulation` operation to be used in more than one `Image`.

ROI in MicrobeamManipulation and Image

- `ROI` in `MicrobeamManipulation` and in `Image` has been reworked inline with its movement to the top level `OME` element.

The ROIs used by the `MicrobeamManipulation` as a 'Ref' and stored in `Image` have been moved to the top level. This allows reuse of the ROI in more than one `Image`.

Other changes

- `LogicalChannel` and `ChannelComponent` have been merged into the new `Channel`.
- Additional keys and indexes to enforce valid IDs and References, are listed in the OME:OME element.
- Support for the `AnalysisModuleLibrary` used by the original OME server has been removed.
- Support for the `DisplayOptions` used by the original OME server has been removed.
- Support for the `Region` used by the original OME server has been removed.
- All description elements and attributes have become simple description elements based on 'xsd::string' that preserve white space.
- The direction of all annotation links has been reversed. Now objects in the model link to annotations, not the other way around.
- Where possible values that are singular are now an attribute of an element, and values that can be multiple are child elements of an element.
- Units are now specified in the schema annotations.
- `Pixels` now link to `Channel`.
- It is now possible to define valid Metadata-Only files.
- There are a large number of general changes and updates as part of a major clean up of the types and names used in the schema, to facilitate code generation direct from the schema.

7.1.11 Changes for September 2008

List of the key changes for the September 2008 major release of the ome-xml data model.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

The version number of all schema files is now 1.

This schema uses the new namespace:

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/2008-09/
```

For the OME schema

```
http://www.openmicroscopy.org/Schemas/OME/2008-09/
```

and that the schema file will be located at

```
http://www.openmicroscopy.org/Schemas/OME/2008-09/ome.xsd
```

Overview of Changes

Additional Schema file SA.xsd

This provides support for the new `StructuredAnnotation` objects. The list of annotations is:

```
XmlAnnotation
FileAnnotation
ListAnnotation
LongAnnotation
DoubleAnnotation
StringAnnotation
BooleanAnnotation
TimestampAnnotation
```

Additional Schema file OMERO.xsd

This provides support for the specific use of OMERO in the `StructuredAnnotation` objects. Each OMERO-specific block is stored in an `XmlAnnotation`. It is not strictly part of the model as it provides support only to the OMERO system. The schema will be published to show how we are using structured annotations within OMERO.

Optional and non-optional Objects

- The following objects have been made **optional** to allow the import of the metadata from Bio-Formats and maintain consistency:

```
OME:Image:CreationDate
OME:Image:Pixels:Plane:PlaneTiming:DeltaT
OME:Image:Pixels:Plane:PlaneTiming:ExposureTime
OME:Experiment:ExperimenterRef
OME:Instrument:Microscope
OME:Instrument:Objective:CalibratedMagnification
OME:Instrument:Objective:LensNA
OME:Instrument:Objective:NominalMagnification
OME:Instrument:Objective:WorkingDistance
OME:Instrument:Filter:TransmittanceRange
OME:Instrument:Filter:TransmittanceRange:CutIn
OME:Instrument:Filter:TransmittanceRange:CutOut
OME:Instrument:Filter:TransmittanceRange:Transmittance
OME:StagePosition:PositionX
OME:StagePosition:PositionY
OME:StagePosition:PositionZ
OME:Project:ExperimenterRef
ManufactSpec:Manufacturer
ManufactSpec:Model
```

`ManufactSpec` is the base type for several other objects.

- The following objects have been made **optional in the model** but **NOT in the OMERO database** - a value will be generated on import:

```
OME:Image:Name
OME:Plate:Name
OME:Dataset:Name
OME:Project:Name
OME:Screen:Name
```

- The following objects **are not optional** - but they now support the value ‘Unknown’:

```
OME:Instrument:LightSource:Laser:LaserMedium - Added new UnknownLaserMedia type to union.
OME:Instrument:LightSource:Laser:Type - added Unknown.
OME:Instrument:LightSource:Arc:Type - added Unknown.
OME:Instrument:LightSource:Filament:Type - added Unknown.
OME:Instrument:Microscope:Type - added Unknown.
OME:Instrument:Objective:Correction - added Unknown.
OME:Instrument:Objective:Immersion - added Unknown.
OME:Instrument:Detector:Type - added Unknown.
```

MicrobeamManipulation

- Moving `MicrobeamManipulation` from `Image` to `Experiment` **(held until next release)**

At present `MicrobeamManipulation` is stored directly under an `Image` with a link to an `Experiment`. The structure this produces is too flexible and allow loops of references to be created. It is proposed that we move all the `MicrobeamManipulation` objects to be located under the `Experiment` and replace them in each `Image` with zero or more `MicrobeamManipulationRef` objects. This also allows one `MicrobeamManipulation` operation to be used in more than one `Image`.

- Reworking `ROI` in `MicrobeamManipulation` **(held until next release)**

As a separate issue to above, the `ROI` used by the `MicrobeamManipulation` should stop being a ‘Ref’ to one stored in `Image`. In the current situation, the manipulated ROI is listed along with all the other ROIs. It is not the same though, as the other ROIs could legitimately be adjusted by the user whereas the `MicrobeamManipulation` ROI is fixed and should not be changed after the manipulation. This change is being made at this time as the ROI is already being changed.

New ROI model

The new model has a more powerful ROI model and the display options are now stored as Structured Annotations. Elements added include `LogicalChannelRef`, `ShapeID`, `ROI`, `Shape`, `BasicSvgShape` and several shapes derived from `BasicSvgShape`.

Min/Max/Optional/Required

All elements now have an explicit minOccurs and maxOccurs. All attributes now have a use value ‘optional’ or ‘required’.

PinholeSize type changed

The PinholeSize type has changed from 'xsd:positiveInteger' to 'xsd:float' with units µm.

Added LightEmittingDiode

This element is a stub to act as a placeholder until more values are needed for this type.

Pixels

Pixels can now contain **EITHER** BinData or TiffData - previously the model allowed there to be a mixture of the two, which was not the intended use.

7.1.12 Changes for February 2008

List of the key changes made for the February 2008 major release of the ome-xml data model.

The new major release of the schema has a new namespace and all version numbers are reset to 1. As a major release, any file that validated correctly using the last major release will probably not validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers and writers to understand the changes.

The version number of all schema files is now 1.

This schema uses the new namespace:

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/2008-02/
```

For the OME schema

```
http://www.openmicroscopy.org/Schemas/OME/2008-02/
```

and the schema file is located at

```
http://www.openmicroscopy.org/Schemas/OME/2008-02/ome.xsd
```

Overview of changes

Description (ome.xsd)

The Description element has become a string type. Description is not the correct place for an XML sub-document. It has also an optional reference to XML:lang that allows the language the Description is written in to be recorded.

Description (AnalysisModule.xsd)

`Description` has been removed for `AnalysisModule`. It is now using the version defined in `ome.xsd`.

UniversallyUniqueIdentifier for files (ome.xsd)

To add support for splitting data across multiple OME-TIFF files in a way that will survive file renaming, each file now has a unique identifier. This is a `UUID` attribute on the `OME` node. It is normally optional but is required if you wish to use multi-part files.

Multi-part OME-TIFF file support (ome.xsd)

A TiffData block can now contain a `UUID` element that specifies the file to look in for the tiff image data. The `UUID` element also has an optional attribute `FileName` to point to the name for the file containing the correct data. The `UUID` is required and the `FileName` is strongly recommended. A reader should first open the file pointed to by `FileName`. If it does not contain the correct `UUID` then the reader can either fail or search the current folder for a file that does contain the correct `UUID`. This allows multi-part files to survive renaming.

Removal of Power attributes from Arc and Filament (ome.xsd)

After `Power` was moved into `LightSource` in the 2007-06 V1 release, this attribute was no longer needed in element `Arc` and `Filament`. These Power attributes should have been removed at the time. Any data can be moved to the `Power` in `LightSource`.

Simplifications (ome.xsd)

The types `Correction` and `Immersion` have been bundled into elements of the same name. They were only used once in the schema. This should not affect instance documents, only any schema derived from `ome.xsd`.

Clarification of PixelTypes (ome.xsd)

The type `PixelTypes` has been split into two. It now has `PixelTypes` and `ExtendedPixelTypes`. It was being used in two different ways in previous schemas, once in `Pixels` and once in `OTF`. Now `Pixels` uses `PixelTypes` and `OTF` uses `ExtendedPixelTypes`.

AcquiredPixelsRef to AcquiredPixels (ome.xsd)

The element `AcquiredPixelsRef` has become attribute `AcquiredPixels`. This has been changed to make `AcquiredPixels` consistent with the `DefaultPixels` attribute in the `Image` element. Now both are a `PixelsID`.

Hash of Plane data (ome.xsd)

There is now the option to store a SHA1 hash of a plane's image data. This can be used to detect modification. It is stored as a `HashSHA1` element inside each `Plane`. At present this is the only supported hashing method. Others may be added in future releases.

Files with no significant changes

The following files have no significant changes from the last version released:

- `AnalysisChain.xsd`
- `BinaryFile.xsd`
- `CA.xsd`
- `CLI.xsd`
- `DataHistory.xsd`
- `MLI.xsd`
- `SPW.xsd`
- `STD.xsd`

Note: The file `SPW.xsd` has had an import of `AnalysisModule.xsd` removed. This import was spurious as it was not used in the file.

7.1.13 Changes for June 2007 release 2

List of the key changes made for the minor release updating the June 2007 ome-xml data model. These changes were introduced in September 2007.

The new minor release of the schema has the same namespace (2007-06) and a new version number. As a minor release, any file that validated correctly using the last major release will also validate correctly using this new release. Some files that failed to validate before will now be valid. It is important to update any file readers to understand the changes. File writers will not necessarily need corrected.

- The version number of the `ome.xsd` schema is now 2.
- The version number of the `SPW.xsd` schema is now 2.
- The version number of the `AnalysisModule.xsd` schema is now 2.
- The version number of the `BinaryFile.xsd` schema is now 2.
- The version number of the `CLI.xsd` schema is now 2.
- The version number of the `MLI.xsd` schema is now 2.
- The version number of the `STD.xsd` schema is now 2.

Overview of changes

Schema Element (all files)

The schema element now imports the XML namespace from a specific location. This gets round a problem with using the old XML namespace which appeared with some XML parsers (notably Xerces), causing the schema to not be valid.

Instrument (ome.xsd)

Parts of the `Instrument` object are now optional:

- `LightSource`
- `Detector`
- `Objective`
- `OTF`

Experimenter and ExperimenterType (ome.xsd)

`FirstName`, `LastName`, `Email` and `OMEName` are all now optional. However, to be valid, an `Experimenter` must have **AT LEAST ONE** of these present. `Experimenter` is now a self-contained object and no longer uses `ExperimenterType`, which has been removed.

Description (ome.xsd)

While there has been no real change to the `Description` element, its annotation has been altered to indicate a change in its use. The content model is currently 'ANY', but this is going to change to 'String'. `Description` is no longer the correct place for an XML sub-document - `CustomAttributes` should be used instead.

SerialNumber (ome.xsd)

`SerialNumber`, part of `ManufactSpec` used by `Objective`, `Detector`, `Microscope` and `LightSource`, has been made optional as there are occasions when the writing application does not have this information. While it may be optional, it must be stressed that it should normally be included, and as such the validator will raise a warning (but not an error) if it is not present or is empty.

Regex for LSID / Internal ID (ome.xsd, SPW.xsd, AnalysisModule.xsd)

There has been a minor change to all the regular expression ('xsd:pattern') portions of the IDs, which means they will no longer generate an error on some parsers (notably the Java one). This concerned the way the - (dash) character was specified as part of a URN and is now escaped correctly.

DBLocation, Label and Parent, parts of Element (STD.xsd)

The `DBLocation` is now optional. `Label` and `Parent` have now been added.

Description and Label (STD.xsd)

These now contain a reference to 'XML:lang', which allows the language the `Description` or `Label` is written in to be recorded.

7.1.14 Changes for June 2007

List of the key changes made for the June 2007 release of the ome-xml data model.

While this may not be a complete list of changes in detail, it should cover all the categories of changes applied to the files. Some changes affect the structure of the data model, some only the names used for concepts, and others only the xsd files themselves.

Overview of changes

Copyright changes

The copyright of the files has been extended to recognize the contribution of the University of Wisconsin at Madison.

Minimum specification

A minimum specification for a valid OME-XML file has been defined. Further information, including samples, is available in the *Minimum Specification* section.

Version

The way version information is recorded in the XSD files has been changed to adhere to best practice. For further information see *Version Information*.

ChannelInfo renamed and extended

This has been renamed to `LogicalChannel`. It also now has references to secondary filters for emission and excitation. `IlluminationType` can now be 'NonLinear', and `Mode` now has the catch-all type 'Other'. Settings have been added for PockelCell lasers.

Detector extended

An attribute `Zoom` has been added for use with confocals. Detector `Type` has also been updated; it can now handle 'EM-CCD', 'APD', and 'CMOS' detectors. The EM-CCD detector type meant that an additional attribute was needed called `AmplificationGain`.

DetectorRef extended

A reference to a detector now has the following addition attributes: a `Voltage`; a `ReadOutRate`, which is speed at which the detector can count the pixels; and a `Binning` setting where detector pixels are combined to form a larger pixels set.

Laser extended

Lasers can now be set as the type 'PockelCell'. A `RepetitionRate` has been added. The `FrequencyDoubled` flag has been replaced by the `FrequencyMultiplication` setting which, as an integer, can support more systems. `Laser` now allows any other `LightSource` to be used as the `Pump`. `AuxLightSource` and `AuxLightSourceRef` have been superseded and therefore removed.

Objective extended

This now has attributes that record the `LensNA`, `WorkingDistance`, `ImmersionType`, `NomimalMagnification` (e.g. x60), `CalibratedMagnification` (e.g. x60.034), and `Correction`. The `Correction` attribute records the coating applied to the lens. Possible `Correction` types are 'UV', 'PlanApo', 'PlanFluor', 'SuperFluor', and 'VioletCorrected'. `Magnification` has been superseded and therefore removed.

ObjectiveRef renamed and extended

`ObjectiveRef` is now called `ObjectiveSettingsRef`. `CorrectionCollar`, `Medium`, and `RefractiveIndex` have been added as attributes.

Pixels extended

`Pixels` can now contain either `BinData` or `TiffData` elements to store the pixel data. It can now also contain `Plane` elements. Attributes have been added to hold `PhysicalSizeX`, `PhysicalSizeY`, `PhysicalSizeZ`, `TimeIncrement`, `WaveStart`, and `WaveIncrement`. These were formerly attached to `Image`.

Filter and Filter Sets rewritten

The previous filter model has been replaced.

Images contracted and expanded

`Image` can now contain the following new elements: `PlateRef`, `AcquiredPixelsRef`, `ROI`, and `MicrobeamManipulation`. The attribute `DefaultPixels` is now required. The attributes `PhysicalSizeX`, `PhysicalSizeY`, `PhysicalSizeZ`, `TimeIncrement`, `WaveStart`, and `WaveIncrement` have been moved to `Pixels`.

Features renamed

Throughout the files, `Feature` and `Features` have become `Region` and `Regions`. This allows the naming to be consistent with the region of interest (ROI) model.

Screening Data Model

The new file `SPW.xsd` has been added to hold the Screen/Plate/Well model. With its creation, `Plate` and `Screen` that were in `ome.xsd` have been moved and re-written.

OME-Tiff extensions

The changes necessary to support the OME-Tiff format have been applied, namely the addition of the `TiffData` element to `Pixels` as an alternative option to the `BinData` element.

Internal IDs and LSID changes

The ID types used throughout the OME-XML model are designed to support identifiers in two forms, full LSID format and internal-only format. The formats of these are now enforced in an improved way, using new regular expressions. For further information see *LSID*.

The Image/Experiment Relationship

This has been modified.

Naming conventions

The naming convention used for attribute, element and enumeration names, already partly implemented, is documented below, and existing names have been corrected as necessary.

Enumerations

Enumeration names will normally use UpperCamelCase, and contain letters and numbers and the dash; no spaces and no other punctuation. The '-' is the only permitted punctuation character (so 'e-' is allowed). Only abbreviations in common use in the field should be used.

Attributes

Attribute names will use UpperCamelCase, letters only, no punctuation. Where practical, whole words will be used rather then abbreviations.

Elements

Element names will use UpperCamelCase, letters only, no punctuation. Where practical, whole words will be used rather then abbreviations. Element names will normally be unique across all the OME files and where an element name is reused, this will be for a specific reason, which is outlined in an annotation present everywhere the element is defined.

Miscellaneous corrections

Some spelling mistakes have been corrected. There have been minor changes to the file layouts.

Plane

This has been modified and now contains some of the structures that were in `Image`.

7.1.15 Changes for September 2007

Content from this page is now at *Changes for June 2007 release 2*.

7.1.16 Information on the version structure

All of the schema files released since 2007 have used a new versioning system. This is composed of two parts, Major and Minor.

Major

The major part of a version is contained in the namespace of each schema. This consists of the month and year the schema was first released. The format for the new URI for the namespace will be

```
http://www.openmicroscopy.org/Schemas/[NameSpaceTitle]/[YearAsFourDigits]-
↪[MonthAsTwoDigits]
```

This means the June 2016 major release for the OME schema uses the namespace

```
http://www.openmicroscopy.org/Schemas/OME/2016-06/
```

and that the schema file will be located at

```
http://www.openmicroscopy.org/Schemas/OME/2016-006/ome.xsd
```

There will be a consultation period before each major release. Major releases may include breaking changes.

Minor

In the schema element of the XSD file, there is a version attribute:

```
<xsd:schema xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06"
    ...
    version="1"
    ... >
```

This is used to record the minor version number of the schema. This starts at 1 and counts up in integers (1, 2, 3 etc.) with each minor version.

A new minor version is used for small non-breaking changes. If a change is more significant, or would break application relying on the current schema, then it must be a major version.

Minor versions will be released as needed.

Examples and consequences

The relationship between a major and minor version has been set up as follows. Any file written by an application conforming to a major version, will always continue to be valid under any minor version update to the schema. The first releases to make use of the major and minor system were June 2007(Major) and September 2007(Minor). If microscope application X saves a file using the major June 2007 version of OME-XML, then that file can be read by any application that understands either the June 2007 major version or the September 2007 minor version. The main consequence of this approach is that file readers have to be kept up to date or have to be written in a way that allows some flexibility.

An example of a minor change is the serial number in the manufacture specification. In the June 2007 major version, a SerialNumber is required in ManafactSpec. It was decided to make this optional as it was not always available to a file writer. This is a minor change as the new file can still validate whether it is present or absent.

An example of a major change is the moving of ImageRef from the SPW.xsd file to the ome.xsd file. This change had been proposed as it is envisaged that in the future ImageRef will be used outside the Screen/Plate/Well model, so it belongs in the same namespace as Image. As any reference to SPW:ImageRef would then have to become OME:ImageRef, this causes validation errors in files following the June 2007 schema and is therefore a new major version.

7.1.17 Technical schema descriptions

Auto-generated documentation is available for each release of the schema, including information on each attribute and element. These are published as XSD files on the OME website. They are usually read by XML validators and parsers but are viewable as text files. Alternatively, you can browse the current version of the entire Schema online.

Transforms are available which convert between the the different versions of the schemas.

INDEX