Workshop

Annotations

- While discussing map annotations, it was noted that there was a need to store T/CO2 (currently in ImagingEnvironment) at a much finer granularity than provided for by the model.
- Some formats store this per timepoint or even per plane, and the ability to specify the ImagingEnvironment per timepoint or per plane would be useful
- More generally, being able to attach an arbitrary annotation at this level of granularity would be generally useful, particularly for vendor-specific acquisition metadata
- Some discussion regarding vendor-specific additions. The general consensus was that DICOM-style private fields were not generally the way to go due to being opaque and making the files largely impenetrable and unusable by end users, and that the ability to use vendor namespaces for custom annotations would likely be sufficient.
- [Richard Myers later noted the need to store at an even finer granularity than plane for e.g. electrophysiology data being acquired at an even higher rate]

ROIs

- Distinguish between ROIs created prior to acquisition and those created by downstream analysis [namespace??]
- Hierarchical ROIs describing, e.g. for cell contents nucleus/nucleoli etc.
- Visible flag on folders
- Visible flag on ROIs (just removed!...)
- [Or maybe store visibility separately from Folders/ROIs as a separate annotation?]
- 3D ROI support
    - Existing 3D ROI specifications mentioned in the lightning talks by Paul Korir
    - 3D volume - voxels i.e. 3D Mask
    - Outline - contours (splines? polylines?)
    - Meshes
    - Geometric shape primitives
    - Using: Microdimensions, 3DVIEW - specs available??
- [Caching of orthogonal planes in OMERO]; creation of 2D ROIs on XZ or YZ planes
- ROIs could ignore tracking--delegate elsewhere
- Splines
- Annotatable custom ROI (like GenericLightSource)

Unconference

Reader integration/decentralisation

- Currently all readers are specified in a central file, and the development process is consequently centralised
- 3i - native slidebook reader integration; class file is in the same jar as a collection of native DLLs
- Making Bio-Formats pluggable: "discovery" of readers with annotations
- Splitting readers.txt to allow addition of supplementary readers within the existing framework could potentially add some needed flexibility while avoiding large scale changes and potential breakage
- There is a need for specifying that extra third-party jars require loading; currently restricted to slidebook
- Installation consistency. The same set of third-party jars may be required to be deployed across all installations of Bio-Formats, Fiji, Insight/Importer, OMERO server in order for correct functioning of third-party additions, e.g. import would require the reader both client- and server-side.
- Plugins do not yet stand fully alone
- Richard Myers: Similar to the "late loading" of DLLs with C++/Windows
- Three options:
    - Full dynamic discovery
    - Use master list, but don't complain about missing (optional) readers
    - No change
- Woolz might also benefit from being split out as for slidebook
- CUDA and other jars also bundle native DLLs

(Design issue now opened - https://github.com/openmicroscopy/design/issues/42)

Localisation (PALM/STORM)

- Processing steps
    - Acquired movie/timeseries (image data)
    - Localisation data (CSV/text)
    - Processed image (image data)
- Currently there is no provision for storing the intermediate tabular localisation data.
- All current vendor solutions are incompatible; the localisation data is not compatible between vendors and no solution has yet become standardised
- Both the acquired movie/timeseries and the localisation data require storing so that the final image data may be reprocessed
- Nikon, Zeiss, Brucker provide all-in-one solutions with fitting.
- ND2 uses separate files for the localisation data

- CZI contains all the data in the same file
- Some users use Matlab and/or ImageJ for visualisation (of the localisation data?) and/or processing
- OMERO.tables might be a potential means of storing the localisation data, but needs exposing at the model/bioformats level to allow seamless extraction from the vendor format upon import
- A CSVAnnotation in the data model might be one way to provide this data in a generic form for such a workflow