Bio-Formats

Release 8.5.0

The Open Microscopy Environment

Sep 10, 2026

CONTENTS

1 About Bio-Formats **3**
- 1.1 License, copyright, and citation . . . 3
- 1.2 Help . . . 3
- 1.3 Bio-Formats versions . . . 4
- 1.4 Why Java? . . . 5
- 1.5 Bio-Formats metadata processing . . . 5

2 User Information **7**
- 2.1 Using Bio-Formats with ImageJ and Fiji . . . 7
- 2.2 Command line tools . . . 20
- 2.3 Using Bio-Formats in MATLAB/Octave . . . 35
- 2.4 Third-party applications . . . 37

3 Developer Documentation **39**
- 3.1 Introduction to Bio-Formats . . . 39
- 3.2 Using Bio-Formats as a Java library . . . 48
- 3.3 Using Bio-Formats as a native C++ library . . . 73
- 3.4 Contributing to Bio-Formats . . . 73
- 3.5 Issue tracking . . . 91

4 Formats **93**
- 4.1 Dataset Structure Table . . . 93
- 4.2 Supported Formats . . . 99
- 4.3 Summary of supported metadata fields . . . 211
- 4.4 Grouping files using a pattern file . . . 446
- 4.5 Support for “precompressed” tiles . . . 448
- 4.6 Additional reader and writer options . . . 448

5 Version history **451**
- 5.1 8.5.0 (2026 March) . . . 451
- 5.2 8.4.0 (2026 January) . . . 452
- 5.3 8.3.0 (2025 August) . . . 453
- 5.4 8.2.0 (2025 May) . . . 455
- 5.5 8.1.1 (2025 March) . . . 455
- 5.6 8.1.0 (2025 February) . . . 456
- 5.7 8.0.1 (2024 November) . . . 457
- 5.8 8.0.0 (2024 October) . . . 457
- 5.9 7.3.1 (2024 July) . . . 458
- 5.10 7.3.0 (2024 April) . . . 459
- 5.11 7.2.0 (2024 February) . . . 460

5.12 7.1.0 (2023 December) . . . 461
5.13 7.0.1 (2023 October) . . . 462
5.14 7.0.0 (2023 August) . . . 464
5.15 6.14.0 (2023 July) . . . 465
5.16 6.13.0 (2023 April) . . . 466
5.17 6.12.0 (2023 February) . . . 468
5.18 6.11.1 (2022 December) . . . 469
5.19 6.11.0 (2022 October) . . . 470
5.20 6.10.1 (2022 August) . . . 470
5.21 6.10.0 (2022 May) . . . 471
5.22 6.9.1 (2022 April) . . . 472
5.23 6.9.0 (2022 February) . . . 473
5.24 6.8.1 (2022 January) . . . 474
5.25 6.8.0 (2021 December) . . . 474
5.26 6.7.0 (2021 August) . . . 475
5.27 6.6.1 (2021 March) . . . 477
5.28 6.6.0 (2020 December) . . . 477
5.29 6.5.1 (2020 July) . . . 479
5.30 6.5.0 (2020 April) . . . 480
5.31 6.4.0 (2020 March) . . . 481
5.32 6.3.1 (2019 December) . . . 482
5.33 6.3.0 (2019 October) . . . 483
5.34 6.2.1 (2019 August) . . . 484
5.35 6.2.0 (2019 July) . . . 485
5.36 6.1.1 (2019 June) . . . 486
5.37 6.1.0 (2019 May) . . . 486
5.38 6.0.1 (2019 March) . . . 488
5.39 6.0.0 (2019 February) . . . 488
5.40 5.9.2 (2018 September 03) . . . 492
5.41 5.9.1 (2018 August 14) . . . 492
5.42 5.9.0 (2018 July 3) . . . 493
5.43 5.8.2 (2018 April 23) . . . 494
5.44 5.8.1 (2018 March 22) . . . 495
5.45 5.8.0 (2018 February 21) . . . 496
5.46 5.7.3 (2018 January 11) . . . 497
5.47 5.7.2 (2017 November 21) . . . 498
5.48 5.7.1 (2017 September 20) . . . 499
5.49 5.7.0 (2017 September 4) . . . 499
5.50 5.6.0 (2017 August 14) . . . 500
5.51 5.5.3 (2017 July 5) . . . 501
5.52 5.5.2 (2017 June 15) . . . 502
5.53 5.5.1 (2017 May 25) . . . 502
5.54 5.5.0 (2017 May 8) . . . 503
5.55 5.4.1 (2017 April 13) . . . 504
5.56 5.4.0 (2017 March 21) . . . 504
5.57 5.3.4 (2017 February 21) . . . 505
5.58 5.3.3 (2017 February 2) . . . 506
5.59 5.3.2 (2017 January 9) . . . 507
5.60 5.3.1 (2016 December 19) . . . 507
5.61 5.3.0 (2016 December 12) . . . 508
5.62 5.2.4 (2016 October 18) . . . 509
5.63 5.2.3 (2016 October 5) . . . 509
5.64 5.2.2 (2016 September 13) . . . 510
5.65 5.2.1 (2016 August 25) . . . 510

5.66 5.2.0 (2016 August 18) . . . 511
5.67 5.1.10 (2016 May 9) . . . 514
5.68 5.1.9 (2016 April 14) . . . 514
5.69 5.1.8 (2016 February 15) . . . 515
5.70 5.1.7 (2015 December 7) . . . 516
5.71 5.1.6 (2015 November 16) . . . 517
5.72 5.1.5 (2015 October 12) . . . 517
5.73 5.1.4 (2015 September 7) . . . 518
5.74 5.1.3 (2015 July 21) . . . 520
5.75 5.1.2 (2015 May 28) . . . 521
5.76 5.1.1 (2015 April 28) . . . 522
5.77 5.1.0 (2015 April 2) . . . 522
5.78 5.0.8 (2015 February 10) . . . 524
5.79 5.0.7 (2015 February 5) . . . 524
5.80 5.0.6 (2014 November 11) . . . 525
5.81 5.0.5 (2014 September 23) . . . 525
5.82 5.0.4 (2014 September 3) . . . 525
5.83 5.0.3 (2014 August 7) . . . 525
5.84 5.0.2 (2014 May 28) . . . 525
5.85 5.0.1 (2014 Apr 7) . . . 526
5.86 5.0.0 (2014 Feb 25) . . . 526
5.87 5.0.0-RC1 (2013 Dec 19) . . . 526
5.88 5.0.0-beta1 (2013 June 20) . . . 527
5.89 4.4.10 (2014 Jan 15) . . . 527
5.90 4.4.9 (2013 Oct 16) . . . 527
5.91 4.4.8 (2013 May 2) . . . 527
5.92 4.4.7 (2013 April 25) . . . 527
5.93 4.4.6 (2013 February 11) . . . 527
5.94 4.4.5 (2012 November 13) . . . 528
5.95 4.4.4 (2012 September 24) . . . 528
5.96 4.4.2 (2012 August 22) . . . 528
5.97 4.4.1 (2012 July 20) . . . 528
5.98 4.4.0 (2012 July 13) . . . 528
5.99 4.3.3 (2011 October 18) . . . 528
5.100 4.3.2 (2011 September 15) . . . 529
5.101 4.3.1 (2011 July 8) . . . 529
5.102 4.3.0 (2011 June 14) . . . 529
5.103 4.2.2 (2010 December 6) . . . 530
5.104 4.2.1 (2010 November 12) . . . 530
5.105 4.2.0 (2010 July 9) . . . 531
5.106 4.1.1 (2009 December 3) . . . 531
5.107 4.0.1 (2009 June 1) . . . 532
5.108 4.0.0 (2009 March 3) . . . 532
5.109 2008 December 29 . . . 532
5.110 2008 August 30 . . . 533
5.111 2008 July 1 . . . 533
5.112 2008 April 17 . . . 534
5.113 2008 Feb 12 . . . 534
5.114 2008 Feb 8 . . . 534
5.115 2008 Feb 1 . . . 534
5.116 2007 Dec 28 . . . 535
5.117 2007 Dec 1 . . . 535
5.118 2007 Nov 21 . . . 535
5.119 2007 Oct 17 . . . 536

5.120 2007 Oct 1 . . . 536
5.121 2007 Sept 11 . . . 536
5.122 2007 Aug 1 . . . 537
5.123 2007 July 16 . . . 537
5.124 2007 July 2 . . . 537
5.125 2007 June 18 . . . 537
5.126 2007 June 6 . . . 537
5.127 2007 May 24 . . . 538
5.128 2007 May 2 . . . 538
5.129 2007 Mar 16 . . . 538
5.130 2007 Mar 7 . . . 539
5.131 2007 Feb 28 . . . 539
5.132 2007 Feb 26 . . . 539
5.133 2007 Feb 7 . . . 539
5.134 2007 Jan 5 . . . 540
5.135 2006 Dec 22 . . . 540
5.136 2006 Nov 30 . . . 540
5.137 2006 Nov 2 . . . 540
5.138 2006 Oct 31 . . . 540
5.139 2006 Oct 30 . . . 541
5.140 2006 Oct 27 . . . 541
5.141 2006 Oct 6 . . . 541
5.142 2006 Sep 27 . . . 542
5.143 2006 Sep 6 . . . 542
5.144 2006 Mar 31 . . . 542

6 Reporting a bug **543**
6.1 Before filing a bug report . . . 543
6.2 Common issues to check . . . 543
6.3 Sending a bug report . . . 544

Index **545**

The following documentation is split into four parts. *About Bio-Formats* explains the goal of the software, discusses how it processes metadata, and provides other useful information such as version history and how to report bugs. *User Information* focuses on how to use Bio-Formats as a plugin for ImageJ and Fiji, and also gives details of other software packages which can use Bio-Formats to read and write microscopy formats. *Developer Documentation* covers more indepth information on using Bio-Formats as a Java library and how to interface from non-Java codes. Finally, *Formats* is a guide to all the file formats currently supported by Bio-Formats.

- *About Bio-Formats*
- *User Information*
- *Developer Documentation*
- *Formats*
- *Version history*
- *Reporting a bug*

Bio-Formats 8.5.0 requires Java 8 or above and uses the *June 2016* release of the OME Model.

Bio-Formats is a community project and we welcome your input. You can find guidance on *Reporting a bug* and contact us via the forums. Further information about how the OME team works and how you can contribute to our projects is in the Contributing Developer Documentation.

CHAPTER

ONE

ABOUT BIO-FORMATS

Bio-Formats is a standalone Java library for reading and writing life sciences image file formats. It is capable of parsing both pixels and metadata for a large number of formats, as well as writing to several open file formats.

The primary goal of Bio-Formats is to facilitate the exchange of microscopy data between different software packages and organizations. It achieves this by converting proprietary microscopy data into an open standard called the OME data model, particularly into the OME-TIFF file format.

We believe the standardization of microscopy metadata to a common structure is of vital importance to the community. You may find LOCI's article on open source software in science of interest.

1.1 License, copyright, and citation

Licensing information for Bio-Formats is described on the OME licensing page. Unless otherwise noted, the current copyright statement is:

Copyright (C) 2005 - 2026 Open Microscopy Environment:

- Board of Regents of the University of Wisconsin-Madison
- Glencoe Software, Inc.
- University of Dundee

Bio-Formats can be cited in publications as follows:

> Melissa Linkert, Curtis T. Rueden, Chris Allan, Jean-Marie Burel, Will Moore, Andrew Patterson, Brian Loranger, Josh Moore, Carlos Neves, Donald MacDonald, Aleksandra Tarkowska, Caitlin Sticco, Emma Hill, Mike Rossner, Kevin W. Eliceiri, and Jason R. Swedlow (2010) Metadata matters: access to image data in the real world. The Journal of Cell Biology 189(5), 777-782. doi: 10.1083/jcb.201004104

PMID:20513764

Additional citation information can be found on the main OME citation page.

1.2 Help

There is a *guide for reporting bugs here*.

For help relating to opening images in ImageJ or FIJI or when using the command line tools, refer to the *users documentation*. You can also find tips on common issues with specific formats on the pages linked from the *supported formats table*.

Please contact us if you have any questions or problems with Bio-Formats not addressed by referring to the documentation.

We no longer recommend reporting Bio-Formats issues in these places, but the archives of the following may be useful as a reference:

- OME Forums
- OME Trac
- ome-devel mailing list (searchable using google with 'site:lists.openmicroscopy.org.uk')
- ome-users mailing list (searchable using google with 'site:lists.openmicroscopy.org.uk')
- ImageJ mailing list
- Fiji GitHub Issues
- Confocal microscopy mailing list

1.3 Bio-Formats versions

Since Bio-Formats 5.1.3, Bio-Formats is decoupled from OMERO with its own release schedule rather than being updated whenever a new version of OMERO is released. This change allows for more frequent releases to get fixes out to the community faster. See the *version history* for a list of changes in each release.

1.3.1 Versioning policy

The following set of rules describe the current versioning policy using RFC 2119.

The Bio-Formats API follows strict semantic versioning since Bio-Formats 5.4.0 i.e.:

- The version number MUST take the form X.Y.Z where X, Y, and Z are non-negative integers, and MUST NOT contain leading zeroes. X is the major version, Y is the minor version and Z is the patch version.
- The patch version Z MUST be incremented if only backwards-compatible bug fixes are introduced. A bug fix is defined as an internal change that fixes incorrect behavior.
- The minor version Y MUST be incremented if new, backwards-compatible functionality is introduced to the public API.
- The major version X MUST be incremented when backwards-incompatible changes are introduced to the public API.
- Either the minor version or the major version MUST be incremented if the version of a non-OME/external dependency is updated.

Serialization functionality was implemented as a ReaderWrapper called Memoizer in Bio-Formats 5.0.0 and is exposed to the community via a public API. Since Bio-Format 5.4.0,:

- The minor version Y MUST be incremented if changes are introduced that are not backwards compatible with regard to serialization.
- Serialized memo files written with a previous minor version MAY not be readable by later versions and MAY need to be rewritten.
- Consumers with code relying on Bio-Formats caching stability SHOULD not upgrade the minor version of Bio-Formats version for now.

For format reader fixes and additions, the policy should read as follows:

- The minor version Y MUST be incremented when a new file format reader is introduced.
- The minor version Y MUST be incremented when a non backwards-compatible format bug fix is introduced, e.g. a fix that modifies the core metadata of existing files.
- The patch version Z MUST be incremented if only backwards-compatible format bug fixes are introduced.

See this GitHub issue for further details.

1.4 Why Java?

From a practical perspective, Bio-Formats is written in Java because it is cross-platform and widely used, with a vast array of libraries for handling common programming tasks. Java is one of the easiest languages from which to deploy cross-platform software. In contrast to C++, which has a large number of complex platform issues to consider, and Python, which leans heavily on C and C++ for many of its components (e.g., NumPy and SciPy), Java code is compiled one time into platform-independent byte code, which can be deployed as is to all supported platforms. And despite this enormous flexibility, Java manages to provide time performance nearly equal to C++, often better in the case of I/O operations (see further discussion on the comparative speed of Java on the LOCI site).

There are also historical reasons associated with the fact that the project grew out of work on the VisAD Java component library. You can read more about the origins of Bio-Formats on the LOCI Bio-Formats homepage.

1.5 Bio-Formats metadata processing

Pixels in microscopy are almost always very straightforward, stored on evenly spaced rectangular grids. It is the metadata (details about the acquisition, experiment, user, and other information) that can be complex. Using the OME data model enables applications to support a single metadata format, rather than the multitude of proprietary formats available today.

Every file format has a distinct set of metadata, stored differently. Bio-Formats processes and converts each format's metadata structures into a standard form called the OME data model, according to the OME-XML specification. We have defined an open exchange format called OME-TIFF that stores its metadata as OME-XML. Any software package that supports OME-TIFF is also compatible with the dozens of formats listed on the Bio-Formats page, because Bio-Formats can convert your files to OME-TIFF format.

To facilitate support of OME-XML, we have created a library in Java for reading and writing OME-XML metadata.

There are three types of metadata in Bio-Formats, which we call core metadata, original metadata, and OME metadata.

1. **Core metadata** only includes things necessary to understand the basic structure of the pixels: image resolution; number of focal planes, time points, channels, and other dimensional axes; byte order; dimension order; color arrangement (RGB, indexed color or separate channels); and thumbnail resolution.
2. **Original metadata** is information specific to a particular file format. These fields are key/value pairs in the original format, with no guarantee of cross-format naming consistency or compatibility. Nomenclature often differs between formats, as each vendor is free to use their own terminology.
3. **OME metadata** is information from #1 and #2 converted by Bio-Formats into the OME data model. **Performing this conversion is the primary purpose of Bio-Formats.** Bio-Formats uses its ability to convert proprietary metadata into OME-XML as part of its integration with the OME and OMERO servers— essentially, they are able to populate their databases in a structured way because Bio-Formats sorts the metadata into the proper places. This conversion is nowhere near complete or bug free, but we are constantly working to improve it. We would greatly appreciate any and all input from users concerning missing or improperly converted metadata fields.

CHAPTER

TWO

USER INFORMATION

Every release of Bio-Formats includes a plugin for ImageJ/Fiji, a set of command line utilities and a MATLAB toolbox.

2.1 Using Bio-Formats with ImageJ and Fiji

The following sections explain the features of Bio-Formats and how to use it within ImageJ and Fiji:

2.1.1 ImageJ overview

ImageJ is an image processing and analysis application written in Java, widely used in the life sciences fields, with an extensible plugin infrastructure. You can use Bio-Formats as a plugin for ImageJ to read and write images in the formats it supports.

Installation

Download bioformats_package.jar and drop it into your **ImageJ/plugins** folder. Next time you run ImageJ, a new Bio-Formats submenu with several plugins will appear in the Plugins menu, including the Bio-Formats Importer and Bio-Formats Exporter.

Usage

The Bio-Formats Importer plugin can display image stacks in several ways:

- In a standard ImageJ window (including as a hyperstack)
- With Joachim Walter's Image5D plugin (if installed)
- With Rainer Heintzmann's View5D plugin (if installed)

ImageJ v1.37 and later automatically (via `HandleExtraFileTypes`) calls the Bio-Formats logic, if installed, as needed when a file is opened within ImageJ, i.e. when using *File → Open* instead of explicitly choosing *Plugins → Bio-Formats → Bio-Formats Importer* from the menu.

For a more detailed description of each plugin, see the Bio-Formats page of the ImageJ wiki.

Upgrading

To upgrade, just overwrite the old **bioformats_package.jar** with the latest one.

You can also upgrade the Bio-Formats plugin directly from ImageJ. Select *Plugins → Bio-Formats → Update Bio-Formats Plugins* from the ImageJ menu, then select which release you would like to use. You will then need to restart ImageJ to complete the upgrade process.

Macros and plugins

Bio-Formats is fully scriptable in a macro, and callable from a plugin. To use in a macro, use the Macro Recorder to record a call to the Bio-Formats Importer with the desired options. You can also perform more targeted metadata queries using the Bio-Formats macro extensions.

Here are some example ImageJ macros and plugins that use Bio-Formats to get you started:

basicMetadata.txt - A macro that uses the Bio-Formats macro extensions to print the chosen file's basic dimensional parameters to the Log.

planeTimings.txt - A macro that uses the Bio-Formats macro extensions to print the chosen file's plane timings to the Log.

recursiveTiffConvert.txt - A macro for recursively converting files to TIFF using Bio-Formats.

bfOpenAsHyperstack.txt - This macro from Wayne Rasband opens a file as a hyperstack using only the Bio-Formats macro extensions (without calling the Bio-Formats Importer plugin).

zvi2HyperStack.txt - This macro from Sebastien Huart reads in a ZVI file using Bio-Formats, synthesizes the LUT using emission wavelength metadata, and displays the result as a hyperstack.

dvSplitTimePoints.txt - This macro from Sebastien Huart splits timepoints/channels on all DV files in a folder.

batchTiffConvert.txt - This macro converts all files in a directory to TIFF using the Bio-Formats macro extensions.

Read_Image - A simple plugin that demonstrates how to use Bio-Formats to read files into ImageJ.

Mass_Importer - A simple plugin that demonstrates how to open all image files in a directory using Bio-Formats, grouping files with similar names to avoiding opening the same dataset more than once.

Usage tips

- "How do I make the options window go away?" is a common question. There are a few ways to do this:
  - To disable the options window only for files in a specific format, select *Plugins* > *Bio-Formats* > *Bio-Formats Plugins Configuration*, then pick the format from the list and make sure the "Windowless" option is checked.
  - To avoid the options window entirely, use the *Plugins* > *Bio-Formats* > *Bio-Formats Windowless Importer* menu item to import files.
  - Open files by calling the Bio-Formats importer plugin from a macro.
- A common cause of problems having multiple copies of `bioformats_package.jar` in you ImageJ plugins folder, or a copy of `bioformats_package.jar` and a copy of `formats-gpl.jar`. It is often difficult to determine for sure that this is the problem - the only error message that pretty much guarantees it is a `NoSuchMethodException`. If you downloaded the latest version and whatever error message or odd behavior you are seeing has been reported as fixed, it is worth removing all copies of `bioformats_package.jar` (or any other Bio-Formats jars) and download a fresh version.
- The Bio-Formats Exporter plugin's file chooser will automatically add the first listed file extension to the file name if a specific file format is selected in the `Files of Type` box (e.g. `.ome.tif` for OME-TIFF). This can prevent BigTIFF and OME BigTIFF files from being created, as the `.btf` or `.ome.btf` file extension will be overwritten. To ensure that the desired extension is used, select *All files* or *All supported file types* in the `Files of type` box, as an extension will not be automatically added in those cases.
- Saving an open image using Bio-Formats must be done via *Plugins* > *Bio-Formats* > *Bio-Formats Exporter* or the corresponding macro code. *File* > *Save* and *File* > *Save As...* do not use Bio-Formats. In particular, using *File* > *Save As...* to save a TIFF will result in an ImageJ-specific TIFF being written. While Bio-Formats can read ImageJ TIFFs, other software may not; see *TIFF (Tagged Image File Format)* for additional information.

2.1.2 Fiji overview

Fiji is an image processing package. It can be described as a distribution of *ImageJ* together with Java, Java 3D and a lot of plugins organized into a coherent menu structure. Fiji compares to ImageJ as Ubuntu compares to Linux.

Fiji works with Bio-Formats out of the box, because it comes bundled with the *Bio-Formats ImageJ plugins*.

The Fiji documentation has been combined with the ImageJ wiki; for further details on Bio-Formats in Fiji, see the Bio-Formats ImageJ page.

Upgrading

Upgrading Bio-Formats within Fiji is as simple as invoking the "Update Fiji" command from the Help menu. By default, Fiji even automatically checks for updates every time it is launched, so you will always be notified when new versions of Bio-Formats (or any other bundled plugin) are available.

Manual upgrade

Manually updating your Fiji installation should not be necessary but if you need to do so, the steps are detailed below. Note that although we assume you will be upgrading to the latest release version, all previous versions of Bio-Formats are available from https://downloads.openmicroscopy.org/bio-formats/ so you can revert to an earlier version using this guide if you need to.

1) Fiji must first be fully updated
2) Close Fiji
3) Open the Fiji installation folder (typically named 'Fiji.app')
4) Remove bio-formats_plugins.jar from the 'plugins' sub-folder
5) Remove all of the .jars from the 'jars/bio-formats' sub-folder:
    - ome-jai.jar
    - formats-gpl.jar
    - ome-common.jar
    - turbojpeg.jar
    - ome-xml.jar
    - formats-bsd.jar
    - ome-poi.jar
    - specification.jar
    - mdbtools-java.jar
    - metakit.jar
    - formats-api.jar
6) Download bio-formats_plugins.jar (from the latest release https://downloads.openmicroscopy.org/bio-formats/) and place it in the 'plugins' sub-folder
7) Download each of the following (from the latest release https://downloads.openmicroscopy.org/bio-formats/) and place them in the 'jars/bio-formats' sub-folder:
    - ome-jai.jar
    - formats-gpl.jar
    - ome-common.jar

- turbojpeg.jar
- ome-xml.jar
- formats-bsd.jar
- ome-poi.jar
- specification.jar
- mdbtools-java.jar
- metakit.jar
- formats-api.jar

8) To Check Version of Bio-Formats *Select Help* > *About Plugins* > *Bio-Formats Plugins...* Check that the version of Bio-Formats matches the freshly downloaded version.

9) Start Fiji and open any Image file using *Plugins* > *Bio-Formats* > *Bio-Formats Importer*

Note

It is vital to perform all of those steps in order; omitting even one will cause a problem. In particular, make sure that the old files are fully removed; it is not sufficient to add the new files to any sub-directory without removing the old files first.

2.1.3 Bio-Formats features in ImageJ and Fiji

When you select Bio-Formats under the Plugin menu, you will see the following features:

- The **Bio-Formats Importer** is a plugin for *loading images* into ImageJ or Fiji. It can read over 140 proprietary life sciences formats and standardizes their acquisition metadata into the common *OME data model*. It will also extract and set basic metadata values such as spatial calibration if they are available in the file.
- The **Bio-Formats Exporter** is a plugin for exporting data to disk. It can save to the open OME-TIFF file format, as well as several movie formats (e.g. QuickTime, AVI) and graphics formats (e.g. PNG, JPEG).
- The **Bio-Formats Remote Importer** is a plugin for importing data from a remote URL. It is likely to be less robust than working with files on disk, so we recommend downloading your data to disk and using the regular Bio-Formats Importer whenever possible.
- The **Bio-Formats Windowless Importer** is a version of the Bio-Formats Importer plugin that runs with the last used settings to avoid any additional dialogs beyond the file chooser. If you always use the same import settings, you may wish to use the windowless importer to save time (Learn more *here*).
- The **Bio-Formats Macro Extensions** plugin prints out the set of commands that can be used to create macro extensions. The commands and the instructions for using them are printed to the ImageJ log window.
- The **Stack Slicer** plugin is a helper plugin used by the Bio-Formats Importer. It can also be used to split a stack across channels, focal planes or time points.
- The **Bio-Formats Plugins Configuration** dialog is a useful way to configure the behavior of the Bio-Formats plugin or each file format. The general tab allows you to configure features of the Bio-Formats plugin such as the display of the slice label (see *Bio-Formats plugin configuration options*). The Formats tab lists supported file formats and toggles each format on or off, which is useful if your file is detected as the wrong format. It also toggles whether each format bypasses the importer options dialog through the "Windowless" checkbox. You can also configure any specific option for each format (see *Additional reader and writer options*). The Libraries tab provides a list of available helper libraries used by Bio-Formats.

- The **Bio-Formats Plugins Shortcut Window** opens a small window with a quick-launch button for each plugin. Dragging and dropping files onto the shortcut window opens them quickly using the **Bio-Formats Importer** plugin.
- The **Update Bio-Formats Plugins** command will check for updates to the plugins. We recommend you update to the newest Trunk build as soon as you think you may have *discovered a bug*.

2.1.4 Installing Bio-Formats in ImageJ

> **Note**
>
> Since FIJI is essentially ImageJ with plugins like Bio-Formats already built in, people who install Fiji can skip this section.
>
> If you are also using the OMERO plugin for ImageJ, you may find the set-up guide on the new user help site useful for getting you started with both plugins at the same time.

Once you download and install ImageJ, you can install the Bio-Formats plugin by going to the Bio-Formats download page and saving the **bioformats_package.jar** to the Plugins directory within ImageJ.

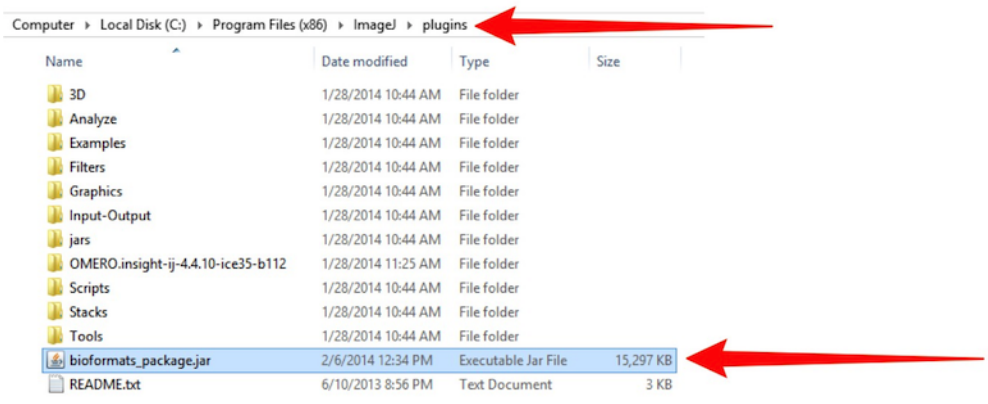
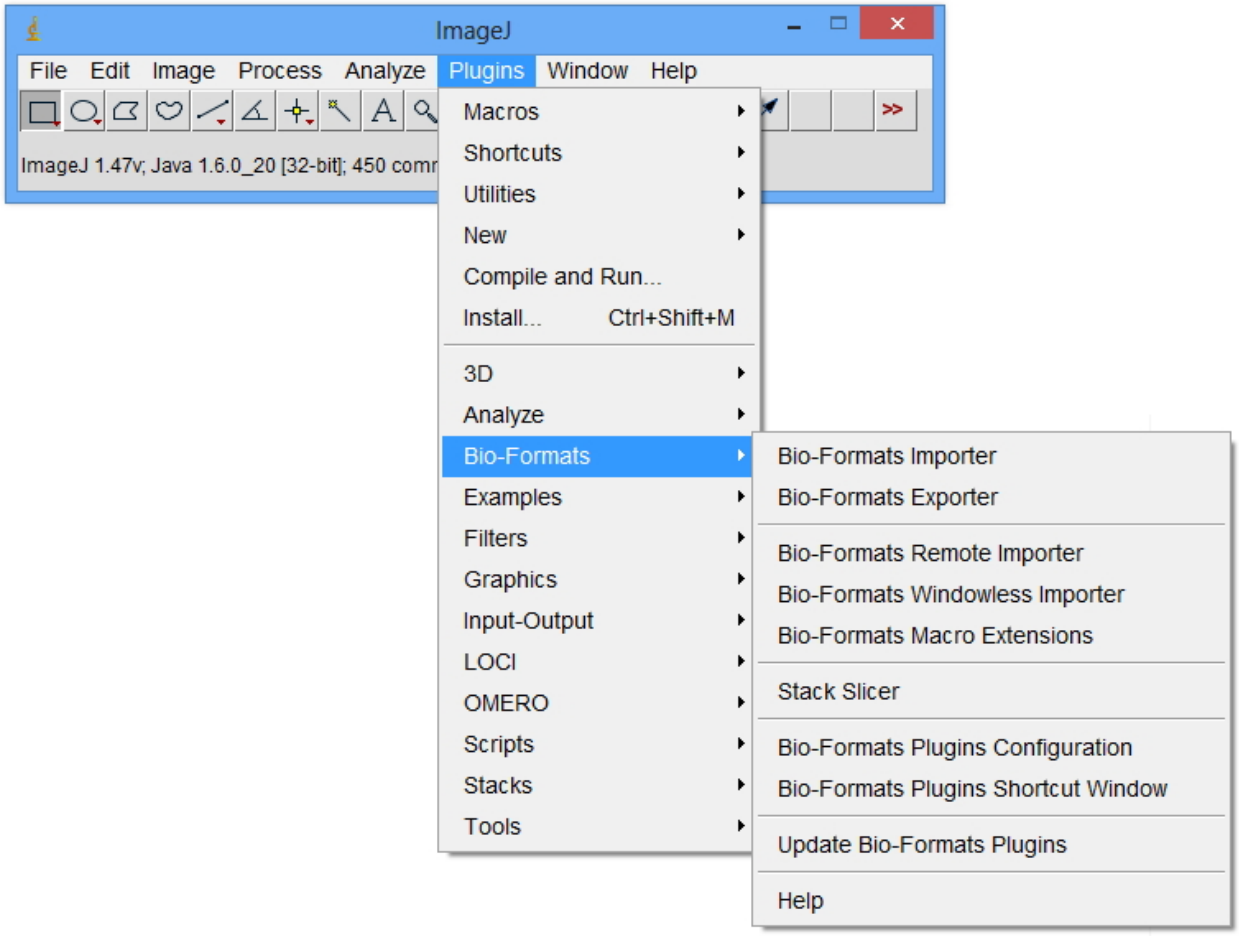


Fig. 1: Plugin Directory for ImageJ: Where in ImageJ's file structure you should place the file once you downloaded it.

You may have to quit and restart ImageJ. Once you restart it, you will find Bio-Formats in the Bio-Formats option under the Plugins menu:



You are now ready to start using Bio-Formats.

2.1.5 Using Bio-Formats to load images into ImageJ

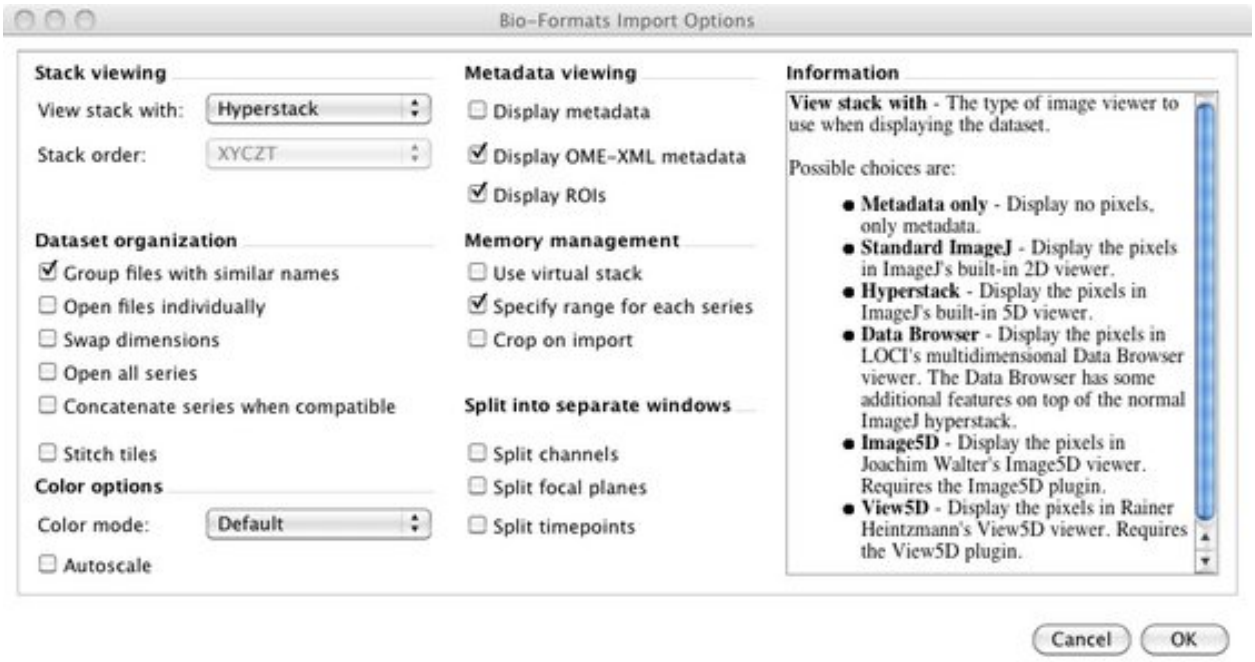
This section will explain how to use Bio-Formats to import files into ImageJ and how to use the settings on the Bio-Formats Import Options screen.

Opening files

There are three ways you can open a file using Bio-Formats:

1. Select the Bio-Formats Importer under the Bio-Formats plugins menu.
2. Drag and drop it onto the Bio-Formats Plugins Shortcut window.
3. Use the Open command in the File menu.

Unless you used the Bio-Formats Plugins Configuration dialog to open the file type windowlessly, you know you used Bio-Formats to open a file when you see a screen like this:



If you used the File > Open command and did not see the Bio-Formats Import Options screen, ImageJ/Fiji probably used another plugin instead of Bio-Formats to open the file. If this happens and you want to open a file using Bio-Formats, use one of the other two methods instead.

Opening files windowlessly

When you open a file with Bio-Formats, the Import Options Screen automatically recalls the settings you last used to open a file with that specific format (e.g. JPG, TIF, LSM, etc.). If you always choose the same options whenever you open files in a specific file format, you can save yourself time by bypassing the Bio-Formats Import Options screen. You can accomplish this two ways:

1. You can select the **Bio-Formats Windowless Importer**, located in the Bio-Formats menu under ImageJ's Plugin menu. When you select this option, Bio-Formats will import the file using the same settings you used the last time you imported a file with the same format.

2. If you invariably use the same settings when you open files in a specific format, you can always bypass the Import Options Screen by changing the settings in the **Bio-Formats Plugins Configuration** option, which is also located in the Bio-Formats menu under ImageJ's Plugin menu.

Once you select this option, select the file format you are interested in from the list on the left side of the screen. Check both the **Enabled** and **Windowless** boxes. Once you do this, whenever you open a file using the **Bio-Formats Windowless Importer,**the **Bio-Formats Importer**, or the drag-and-drop method described in the previous section, the file will always open the same way using the last setting used.

Please note that if you want to change any of the import settings once you enable this windowless option, you will have to go back to the **Bio-Formats Plugins Configuration** screen, unselect the windowless option, open a file using the regular **Bio-Formats Importer**, select your settings, and re-select the windowless option.

Group files with similar names

Note

The functionality described below is also available outside ImageJ, by using a pattern file to tell Bio-Formats how

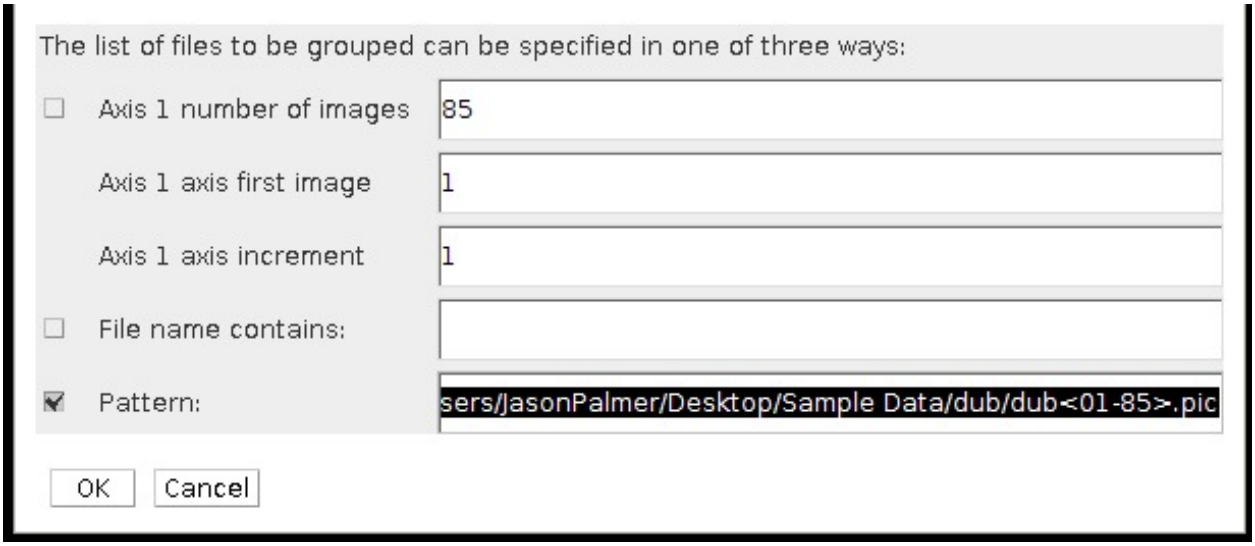
to group the files. See *Grouping files using a pattern file* for more information.

One of the most important features of Bio-Formats is to combine multiple files from a data set into one coherent, multi-dimensional image.

To demonstrate how to use the **Group files with similar names** feature, you can use the *dub* data set available under LOCI's Sample Data page. You will notice that it is a large dataset: each of the 85 files shows the specimen at 33 optical sections along the z-plane at a specific time.

If you open just one file in ImageJ/Fiji using the **Bio-Formats Importer**, you will get an image incorporating three dimensions (x, y, z). However, if you select **Group files with similar names** from the Bio-Formats Import Options screen, you will be able to create a 4-D image (x, y, z, and t) incorporating the 85 files.

After clicking OK, you will see a screen like this:



This screen allows you to select which files within the 85-file cluster to use to create that 4-D image. Some information will be pre-populated in the fields. Unless you want to change the settings in that field, there is no need to change or delete it. If you click OK at this point, you will load all 85 files.

However, you can specify which files you want to open by adjusting the "axis information", the file "name contains", or the "pattern" sections. Even though there are three options, you only need to need to make changes to one of them. Since Bio-Format's precedence for processing data is from top to bottom, only the uppermost section that you made changes to will be used. If you change multiple boxes, any information you enter into lower boxes will be ignored.

To return to the example involving the dub data set, suppose you want to open the first image and only every fifth image afterwards (i.e. dub01, dub06, dub11 … dub81). This would give you 17 images. There are different ways to accomplish this:

You can use the **Axis Settings** only when your files are numbered in sequential order and you want to open only a subset of the files that have similar names. Since the dub data set is numbered sequentially, you can use this feature.

Axis 1 number of images refers to the total number of images you want to open. Since you want to view 17 images, enter 17. **Axis 1 axis first image** specifies which image in the set you want to be the first. Since you want to start with dub01, enter 1 in that box. You also want to view only every fifth image, so enter 5 in the **Axis 1 axis increment** box.

The **File name contains** box should be used if all of the files that you want to open have common text. This is especially useful when the files are not numbered. For example, if you have "Image_Red.tif", "Image_Green.tif", and "Image_Blue.tif" you could enter "Image_" in the box to group them all.

To continue the example involving the dub data set, you cannot use the **file name contains** box to open every fifth image. However, if you only wanted to open dub10 thorough dub19, you could enter "dub1" in the **file name contains** box.

The **pattern** box can be used to do either of the options listed above or much more. This box can accept a single file name like "dub01.pic". It can also contain a pattern that use "<" and ">" to specify what numbers or text the file names contain.

There are three basic forms to the "< >" blocks:

- Text enumeration - "Image_<Red,Green,Blue>.tif" is the pattern for Image_Red.tif, Image_Green.tif, Image_Blue.tif. (Note that the order you in which you enter the file names is the order in which they will be loaded.)
- Number range - "dub<1-85>.pic" is the pattern for "dub1.pic", "dub2.pic", "dub3.pic" … "dub85.pic".
- Number range with step - "dub<1-85:5>.pic" is the pattern for "dub1.pic", "dub6.pic", "dub11.pic", "dub11.pic" … "dub85.pic".

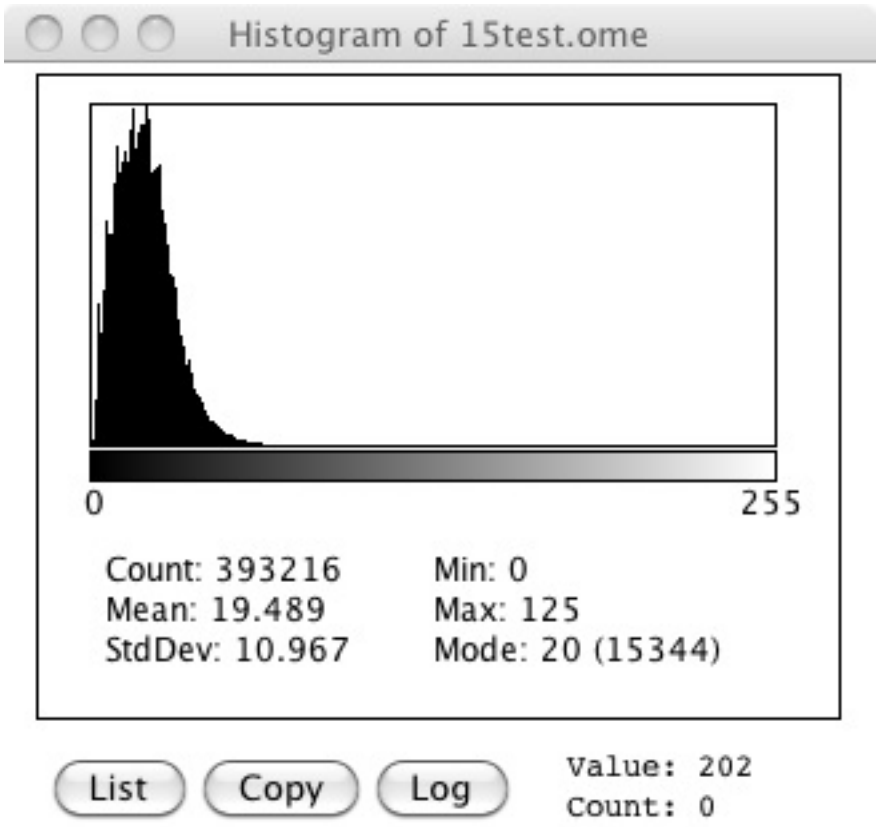
It can also accept a Java regular expression.

Autoscale

Autoscale helps increase the brightness and contrast of an image by adjusting the range of light intensity within an image to match the range of possible display values. Note that Autoscale does not change your data. It just changes how it is displayed.

Each pixel in an image has a numerical value ascribed to it to describe its intensity. The bit depth—the number of possible values– depends on the number of bits used in the image. Eight bits, for example, gives 256 values to express intensity where 0 is completely black, 255 is completely white, and 1 through 254 display increasingly lighter shades of grey.

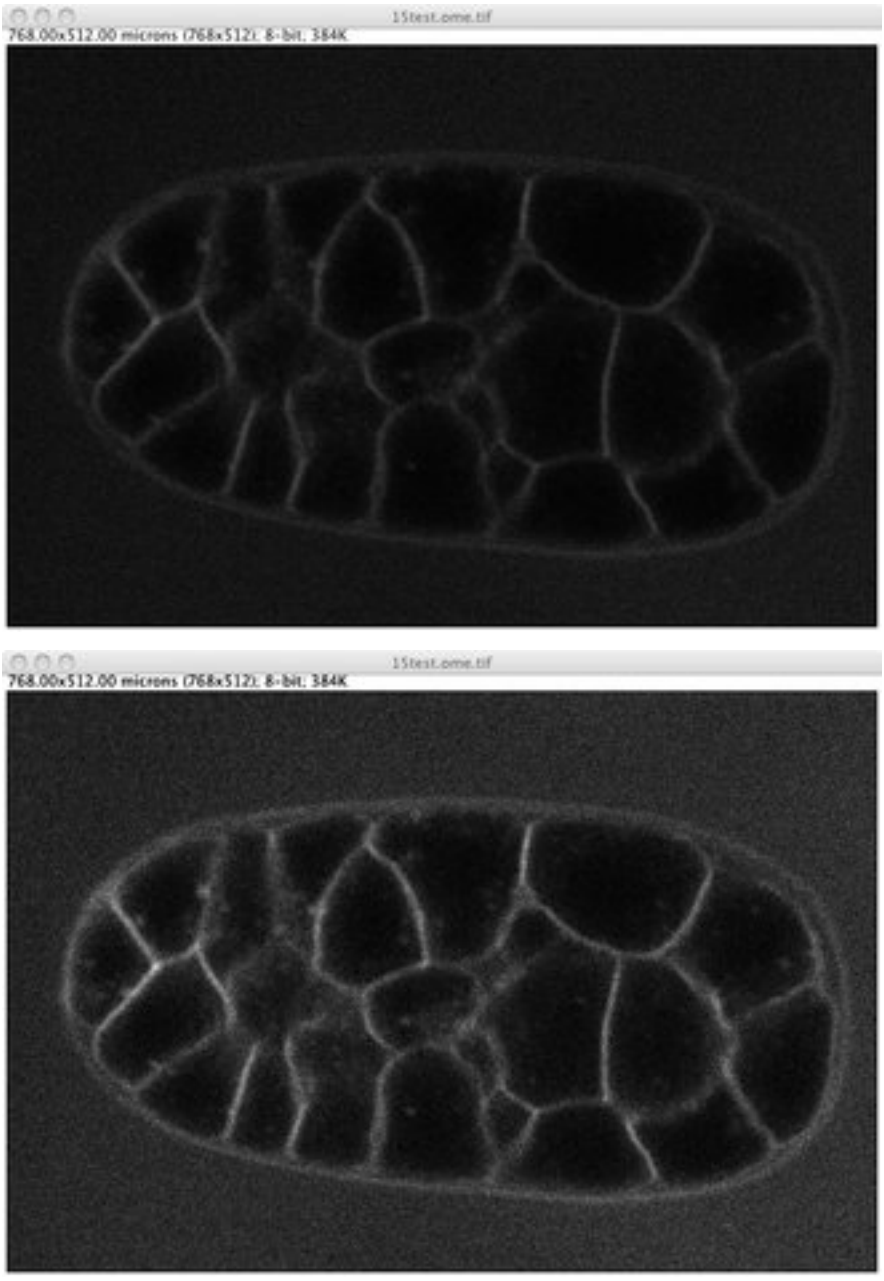
ImageJ can collect the intensity information about each pixel from an image or stack and create a histogram (you can see it by selecting Histogram under the Analyze menu). Here is the histogram of a one particular image:



Notice that the histogram heavily skews left. Even though there are 256 possible values, only 0 thorough 125 are being used.

Autoscale adjusts the image so the smallest and largest number in that image or stack's histogram become the darkest and brightest settings. For this image, pixels with the intensity of 125 will be displayed in pure white. The other values will be adjusted too to help show contrast between values that were too insignificant to see before.

Here is one image Bio-Formats imported with and without using Autoscale:

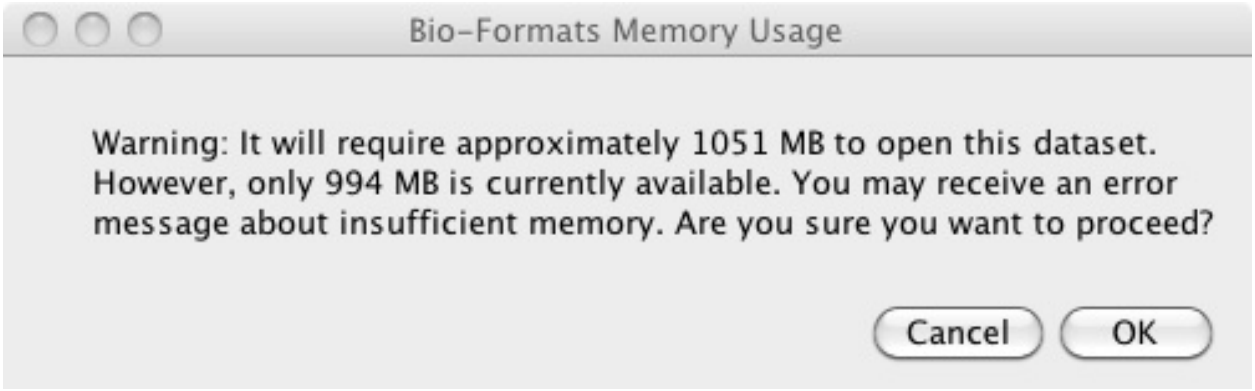


Autoscale readjusts the image based on the highest value in the entire data set. This means if the highest value in your dataset is close to maximum display value, Autoscale's adjusting may be undetectable to the eye.

ImageJ/Fiji also has its own tools for adjusting the image, which are available by selecting Brightness/Contrast, which is under the Adjust option in the Image menu.

2.1.6 Managing memory in ImageJ/Fiji using Bio-Formats

When dealing with a large stack of images, you may receive a warning like this:



This means the allotted memory is less than what Bio-Formats needs to load all the images. If you have a very large data set, you may have to:

- Crop the view area
- Open only a subset of images
- Use Virtual Stack
- Increase ImageJ/Fiji's memory.

If your files contain JPEG or JPEG-2000 images, you may see this memory warning even if your file size is smaller than the amount of allocated memory. This is because compressed images like JPEG need to be decompressed into memory before being displayed and require more memory than their file size suggests. If you are having this issue, try utilizing one of the memory management tools below.

Cropping the view area

Crop on Import is useful if your images are very large and you are only interested in one specific section of the stack you are importing. If you select this feature, you will see a screen where you can enter the height and width (in pixels) of the part of image you want to see. Note that these measurements are from the top left corner of the image.

Opening only a subset of images

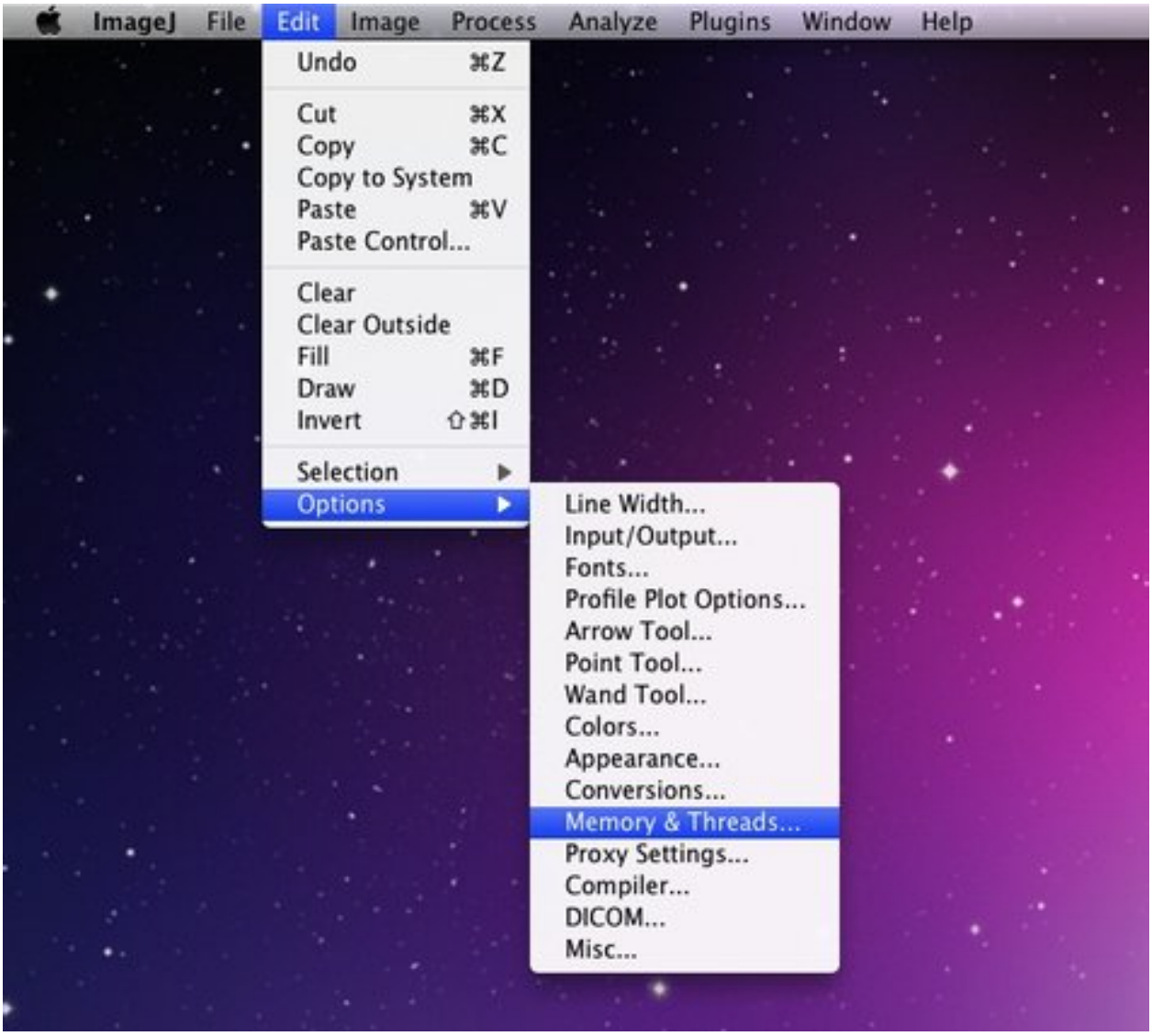
The **Specify Range for Each Series** option is useful for viewing a portion of a data set where all the plane images are encapsulated into one file (e.g. the Zeiss LSM format). If your file has a large quantity of images, you can specify which channels, Z-planes, and times you want to load.

Use Virtual Stack

Virtual Stack conserves memory by not loading specific images until necessary. Virtual Stack does not contain a buffer and may produce choppy animations.

Increasing ImageJ/Fiji's memory

Finally, you can also increase the amount of the computer memory devoted to ImageJ/Fiji by selecting **Memory & Threads** under the **Edit** menu.



Generally, allocating more than 75% of the computer's total memory will cause ImageJ/Fiji to become slow and unstable.

Please note that, unlike the other two features, ImageJ/Fiji itself provides this feature and not Bio-Formats. You can find out more about this feature by looking at ImageJ's documentation.

2.1.7 Bio-Formats plugin configuration options

The Bio-Formats plugin can be configured by opening the **Bio-Formats Plugins Configuration** dialog from the Plugin menu. The General tab allows for configuration of features for the plugin such as upgrade checking or modifying the slice label pattern.

Configuring the slice label pattern

The slice label is the text displayed at the top of the image window in ImageJ. The values displayed here can be modified by configuring the slice label pattern.

The list of available parameters for configuration is as follows::

%s
: series index

%n
: series name

%c
: channel index

%w
: channel name

%z
: Z index

%t
: T index

%A
: acquisition timestamp

Each index value will be 1-based rather than 0-based. The index will be displayed along with the total dimension count and with a prefix for the particular dimension. For example using `%c` for channel index will result in the display `c:1/3`.

See also

Additional reader and writer options

2.2 Command line tools

The Bio-Formats command line tools provide a complete package for carrying out a variety of tasks:

2.2.1 Command line tools overview

There are several scripts for using Bio-Formats on the command line.

Installation

Download bftools.zip, unzip it into a new folder.

Note

As of Bio-Formats 5.0.0, this zip now contains the bundled jar and you no longer need to download `bioformats_package.jar` separately.

The zip file contains both Unix scripts and Windows batch files.

Tools available

Currently available tools include:

showinf
: Prints information about a given image file to the console, and displays the image itself in the Bio-Formats image viewer (see *Displaying images and metadata* for more information).

ijview
: Displays the given image file in ImageJ using the Bio-Formats Importer plugin. See *Display file in ImageJ* for details.

bfconvert
: Converts an image file from one format to another. Bio-Formats must support writing to the output file (see *Converting a file to different format* for more information).

formatlist
: Displays a list of supported file formats in HTML, plaintext or XML. See *List supported file formats* for details.

xmlindent
: A simple XML prettifier similar to **`xmllint --format`** but more robust in that it attempts to produce output regardless of syntax errors in the XML. See *Format XML data* for details.

xmlvalid
: A command-line XML validation tool, useful for checking an OME-XML document for compliance with the OME-XML schema. See *Validating XML in an OME-TIFF* for details.

tiffcomment
: Dumps the comment from the given TIFF file's first IFD entry; useful for examining the OME-XML block in an OME-TIFF file (also see *Editing XML in an OME-TIFF*).

domainlist
: Displays a list of imaging domains and the supported formats associated with each domain. See *List formats by domain* for more information.

mkfake
: Creates a "fake" high-content screen with configurable dimensions. This is useful for testing how HCS metadata is handled, without requiring real image data from an acquired screen. See *Create a high-content screen for testing* for more information.

Some of these tools also work in combination, for example *Validating XML in an OME-TIFF* uses both **`tiffcomment`** and **`xmlvalid`**.

Running any of these commands without any arguments will print usage information to help you. When run with the `-version` argument, **`showinf`** and **`bfconvert`** will display the version of Bio-Formats that is being used (version number, build date, and Git commit reference).

Command-line environment

A set of environment variables can be passed to all command-line utilities:

`BF_CP`
: Extra directories to be added to the autodetected command-line classpath e.g. for external reader JARs. Default: empty.

`BF_FLAGS`
: Additional flags to be sent to the JVM. Default: empty.

`BF_MAX_MEM`
: Maximum heap size to be allocated to the JVM. Default: 512m.

`BF_PROFILE`
: Enable profiling - see *Profiling* for more information. Default: off.

`BF_PROFILE_DEPTH`
: Maximum profiling depth if profiling is activated. Default: 30.

Using the tools directly from source

Firstly, obtain a copy of the sources and build them (see *Obtaining and building Bio-Formats*). You can configure the scripts to use your source tree instead of **bioformats_package.jar** in the same directory by following these steps:

1. Point your CLASSPATH to the checked-out directory and the JAR files in the **jar** folder.
    - E.g. on Windows with Java 1.8 or later, if you have checked out the source at `C:\code\bio-formats`, set your CLASSPATH environment variable to the value `C:\code\bio-formats\jar\*;C:\code\bio-formats`. You can access the environment variable configuration area by right-clicking on My Computer, choosing Properties, Advanced tab, Environment Variables button.
2. Compile the source with `ant compile`.
3. Set the `BF_DEVEL` environment variable to any value (the variable just needs to be defined).

Version checker

If you run bftools outside of the OMERO environment, you may encounter an issue with the automatic version checker causing a tool to crash when trying to connect to `upgrade.openmicroscopy.org.uk`. The error message will look something like this:

```
Failed to compare version numbers
java.io.IOException: Server returned HTTP response code: 400 for URL:
http://upgrade.openmicroscopy.org.uk?version=4.4.8;os.name=Linux;os.
version=2.6.32-358.6.2.el6.x86_64;os.arch=amd64;java.runtime.version=
1.6.0_24-b24;java.vm.vendor=Sun+Microsystems+Inc.;bioformats.caller=
Bio-Formats+utilities
```

To avoid this issue, call the tool with the `-no-upgrade` parameter.

Profiling

For debugging errors or investigating performance issues, it can be useful to use profiling tools while running Bio-Formats. The command-line tools can invoke the HPROF agent library to profile Heap and CPU usage. Setting the `BF_PROFILE` environment variable allows to turn profiling on, e.g.:

```
BF_PROFILE=true showinf -nopix -no-upgrade myfile
```

2.2.2 Displaying images and metadata

The **showinf** *command line tool* can be used to show the images and metadata contained in a file.

If no options are specified, **showinf** displays a summary of available options.

To simply display images:

```
showinf /path/to/file
```

All of the images in the first 'series' (or 5 dimensional stack) will be opened and displayed in a simple image viewer. The number of series, image dimensions, and other basic metadata will be printed to the console.

-noflat

Do not flatten resolutions into individual series:

```
showinf -noflat /path/to/file
```

-series SERIES

Displays a different series, for example the second one:

```
showinf -series 1 /path/to/file
```

Note that series numbers begin with 0.

-omexml

Displays the OME-XML metadata for a file on the console:

```
showinf -omexml /path/to/file
```

-nopix

Image reading can be suppressed if only the metadata is needed:

```
showinf -nopix /path/to/file
```

-option KEY VALUE

Passes options expressed as key/value pairs:

```
showinf -option key value /path/to/file
```

e.g. additional reader options, see *Additional reader and writer options*:

```
showinf -option leicalif.old_physical_size true /path/to/file
```

Added in version 5.3.0.

-range START END

A subset of images can also be opened instead of the entire stack, by specifying the start and end plane indices (inclusive):

```
showinf -range 0 0 /path/to/file
```

That opens only the first image in first series in the file.

-crop X,Y,WIDTH,HEIGHT

For very large images, it may also be useful to open a small tile from the image instead of reading everything into memory. To open the upper-left-most 512×512 tile from the images:

```
showinf -crop 0,0,512,512 /path/to/file
```

The parameter to *-crop* is of the format `x,y,width,height`. The (x, y) coordinate (0, 0) is the upper-left corner of the image; `x + width` must be less than or equal to the image width and `y + height` must be less than or equal to the image height.

-no-upgrade

By default, **showinf** will check for a new version of Bio-Formats. This can take several seconds (especially on a slow internet connection); to save time, the update check can be disabled:

```
showinf -no-upgrade /path/to/file
```

-novalid

Similarly, if OME-XML is displayed then it will automatically be validated. On slow or missing internet connections, this can take some time, and so can be disabled:

```
showinf -novalid /path/to/file
```

-nocore

Most output can be suppressed:

```
showinf -nocore /path/to/file
```

-omexml-only

Displays the OME-XML alone:

```
showinf -omexml-only /path/to/file
```

This is particularly helpful when there are hundreds or thousands of series.

-debug

Enables debugging output if more information is needed:

```
showinf -debug /path/to/file
```

-fast

Displays an image as quickly as possible. This is achieved by converting the raw data into a 8 bit RGB image:

```
showinf -fast /path/to/file
```

> **Note**
>
> Due to the data conversion to a RGB image, using this option results in a loss of precision.

-autoscale

Adjusts the display range to the minimum and maximum pixel values:

```
showinf -autoscale /path/to/file
```

> **Note**
>
> This option automatically sets the *-fast* option and suffers from the same limitations.

-cache

Caches the reader under the same directory as the input file after initialization:

```
showinf -cache /path/to/file
```

-cache-dir DIR

Specifies the base directory under which the reader should be cached:

```
showinf -cache-dir /tmp/cachedir /path/to/file
```

-swap DIMENSIONORDER

Overrides the default input dimension order:

```
showinf -swap XYZTC /path/to/file
```

-format FORMAT

Specifies the reader to be used for opening the specified file. The utility will look for a reader named `loci.formats.in.<FORMAT>Reader`. If the reader does not exist or no *-format* option is passed, the file will be opened with `loci.formats.in.ImageReader`:

```
showinf -format APNG test.png
```

-fill UNSIGNED_BYTE

Sets the fill value to use for undefined pixels. Valid values are 0-255, with 0 representing a black pixel and 255 representing a white pixel. The value set here will be applied to all bytes in an undefined pixel, e.g. setting to 128 (0x80) on a uint16 image will result in a pixel value of 32896 (0x8080). The default fill value is typically 0, but some readers may have different defaults.

Added in version 6.13.0.

-

Batch process multiple files reading file names and any additional options line by line from stdin. For every line from stdin `showinf` is invoked as called from the command prompt, but substituting - with the stdin line.

```
ls *.tiff | showinf -
```

Added in version 8.0.0.

2.2.3 Converting a file to different format

The **bfconvert** *command line tool* can be used to convert files between *supported formats*.

bfconvert with no options displays a summary of available options.

To convert a file to single output file (e.g. TIFF):

```
bfconvert /path/to/input output.tiff
```

The output file format is determined by the extension of the output file, e.g. .tiff for TIFF files, .ome.tiff for OME-TIFF, .png for PNG.

-option KEY VALUE

Passes options expressed as key/value pairs:

```
bfconvert -option key value /path/to/input /path/to/output
```

e.g. additional writer options, see *Additional reader and writer options*:

```
bfconvert -option ometiff.companion converted.companion.ome input.fake converted.
↪ome.tiff
```

Added in version 5.4.0.

-noflat

Do not flatten resolutions into individual series. This option is mandatory to read images with pyramidal levels using the sub-resolution API and generate an output image with sub-resolutions. As of Bio-Formats 6.0.0, only the OME-TIFF output format properly supports this option:

```
bfconvert -noflat /path/to/input output-first-series.ome.tiff
```

Added in version 6.0.0.

-series SERIES

All images in the input file are converted by default. To convert only one series:

```
bfconvert -series 0 /path/to/input output-first-series.tiff
```

-timepoint TIMEPOINT

To convert only one timepoint:

```
bfconvert -timepoint 0 /path/to/input output-first-timepoint.tiff
```

-channel CHANNEL

To convert only one channel:

```
bfconvert -channel 0 /path/to/input output-first-channel.tiff
```

-z Z

To convert only one Z section:

```
bfconvert -z 0 /path/to/input output-first-z.tiff
```

-range START END

To convert images between certain indices (inclusive):

```
bfconvert -range 0 2 /path/to/input output-first-3-images.tiff
```

-tilex TILEX, **-tiley** TILEY

All images larger than 4096×4096 will be saved as a set of tiles if the output format supports doing so. The default tile size is determined by the input format, and can be overridden like this:

```
bfconvert -tilex 512 -tiley 512 /path/to/input output-512x512-tiles.tiff
```

-tilex is the width in pixels of each tile; *-tiley* is the height in pixels of each tile. The last row and column of tiles may be slightly smaller if the image width and height are not multiples of the specified tile width and height. Note that specifying *-tilex* and *-tiley* will cause tiles to be written even if the image is smaller than 4096×4096.

Also note that the specified tile size will affect performance. If large amounts of data are being processed, it is a good idea to try converting a single tile with a few different tile sizes using the *-crop* option. This gives an idea of what the most performant size will be.

The selected compression type can also impact how different tile sizes perform. For example, using *-compression JPEG* with TIFF or OME-TIFF output may result in a relatively large file with default tile sizes, but a smaller file when a smaller tile size is used.

Default tile sizes depend upon a few factors:

- If the input image contains fewer than 4096x4096 pixels, the tile size is the full image size.
- If the input image contains more than 4096x4096 pixels, the tile size is "optimal" tile size of the input image. This size will vary depending upon the input file format and the specific dataset, and may be as large as the entire image. The "optimal" tile size typically reflects either a single compressed tile, or 1MB of uncompressed image data.

- If *-precompressed* is specified, then the tile size is always the optimal tile size of the input image.

-crop X,Y,WIDTH,HEIGHT

For very large images, it may also be useful to convert a small tile from the image instead of reading everything into memory. To convert the upper-left-most 512×512 tile from the images:

```
bfconvert -crop 0,0,512,512 /path/to/file output-512x512-crop.tiff
```

The parameter to *-crop* is of the format `x,y,width,height`. The (x, y) coordinate (0, 0) is the upper-left corner of the image; `x + width` must be less than or equal to the image width and `y + height` must be less than or equal to the image height.

Images can also be written to multiple files by specifying a pattern string in the output file. For example, to write one series, timepoint, channel, and Z section per file:

```
bfconvert /path/to/input output_series_%s_Z%z_C%c_T%t.tiff
```

`%s` is the series index, `%z` is the Z section index, `%c` is the channel index, and `%t` is the timepoint index (all indices begin at 0).

For large images in particular, it can also be useful to write each tile to a separate file:

```
bfconvert -tilex 512 -tiley 512 /path/to/input output_tile_%x_%y_%m.jpg
```

`%x` is the row index of the tile, `%y` is the column index of the tile, and `%m` is the overall tile index. As above, all indices begin at 0. Note that if `%x` or `%y` is included in the file name pattern, then the other must be included too. The only exception is if `%m` was also included in the pattern.

Note for Windows Users: The command interpreter for batch files needs the `%` characters to be doubled in order to process the sequencing variables correctly. So in Windows, the above example would read:

```
bfconvert /path/to/input output_series_%%s_Z%%z_C%%c_T%%t.tif
```

-compression COMPRESSION

By default, all images will be written uncompressed. Supported compression modes vary based upon the output format, but when multiple modes are available the compression can be changed using the *-compression* option. For example, to use LZW compression in a TIFF file:

```
bfconvert -compression LZW /path/to/input output-lzw.tiff
```

-quality QUALITY

Specify the compression quality to be used. The interpretation of this value depends upon the *-compression* option used.

For uncompressed output, *-quality* has no effect.

When used with *-compression JPEG*, *-quality* must take a value between `0.25` (25%, extremely lossy) and `1.0` (100%, nearly lossless). The default is `0.75`.

When used with *-compression JPEG-2000*, *-quality* must take a positive integer value. This is interpreted as an encoding rate in bits per pixel, with higher values representing less lossy compression. The default is `10`.

Added in version 7.3.0.

-overwrite

If the specified output file already exists, **bfconvert** will prompt to overwrite the file. When running **bfconvert** non-interactively, it may be useful to always allow **bfconvert** to overwrite the output file:

```
bfconvert -overwrite /path/to/input /path/to/output
```

-nooverwrite

To always exit without overwriting:

```
bfconvert -nooverwrite /path/to/input /path/to/output
```

-nolookup

To disable the conversion of lookup tables, leaving the output file without any lookup tables:

```
bfconvert -nolookup /path/to/input /path/to/output
```

Added in version 5.2.1.

-bigtiff

This option forces the writing of a BigTiff file:

```
bfconvert -bigtiff /path/to/input output.ome.tiff
```

Added in version 5.1.2.

The *-bigtiff* option is not necessary if a BigTiff extension is used for the output file, e.g.:

```
bfconvert /path/to/input output.ome.btf
```

-nobigtiff

This option disables the automatic switching to BigTiff based upon the number of pixel bytes (TIFF files larger than 4GB):

```
bfconvert -nobigtiff /path/to/input output.ome.tiff
```

Added in version 6.4.0.

Using the *-nobigtiff* will disable writing BigTiff when the output format is less than 4GB. It will not be able to write standard Tiff files greater than 4GB. An example of when it might be used would be when converting using a compression codec that reduces the size of the output file, e.g.:

```
bfconvert -nobigtiff -compression LZW /path/to/input output.ome.btf
```

-padded

This option is used alongside a pattern string when writing an image to multiple files. When set this will enforce zero padding on the filename indexes set in the provided pattern string:

```
bfconvert /path/to/input output_xy%sz%zc%ct%t.ome.tif -padded
```

Added in version 5.2.2.

-pyramid-resolutions RESOLUTIONS

-pyramid-scale SCALE

When using *-noflat* by default, each series of the converted file will contain the same number of resolutions as in the input file. The *-pyramid-resolutions* option allows to set the number of expected resolutions in the output file for each series. If the target number of resolutions is greater than the actual number of sub-resolutions present in the input file, additional pyramidal levels will be calculated using the downsampling factor specified by the *-pyramid-scale* option:

```
bfconvert -noflat -pyramid-resolutions 4 -pyramid-scale 2 /path/to/input out.ome.
↪tiff
```

Added in version 6.0.0.

-cache

This option will cache the initialized reader under the same directory as the input file after initialization:

```
bfconvert -cache /path/to/input output.ome.tiff
```

Added in version 6.2.0.

-cache-dir DIRECTORY

This option is to be used in conjunction with *-cache*. When used it specifies the directory to store the cached initialized reader. If unspecified, the cached reader will be stored under the same folder as the image file:

```
bfconvert -cache-dir /path/to/store/cached/reader /path/to/input output.ome.tiff
```

Added in version 6.2.0.

-no-sas

Do not preserve the OME-XML StructuredAnnotation elements:

```
bfconvert -no-sas /path/to/input output.ome.tiff
```

Added in version 6.2.0.

-no-sequential

Do not assume that planes are written in sequential order:

```
bfconvert -no-sequential /path/to/input output.ome.tiff
```

Added in version 6.8.0.

-swap DIMENSIONORDER

Overrides the default input dimension order:

```
bfconvert -swap XYZTC /path/to/input output.ome.tiff
```

Added in version 6.9.0.

-fill UNSIGNED_BYTE

Sets the fill value to use for undefined pixels. Valid values are 0-255, with 0 representing a black pixel and 255 representing a white pixel. The value set here will be applied to all bytes in an undefined pixel, e.g. setting to 128 (0x80) on a uint16 image will result in a pixel value of 32896 (0x8080). The default fill value is typically 0, but some readers may have different defaults.

Added in version 6.13.0.

-precompressed

Enables transfer of compressed bytes from input dataset directly to output without decompression. Most input and output formats do not support this option, currently this is implemented in the Aperio SVS input and DICOM output. Do not use -crop, -fill, or -autoscale, or pyramid generation options:

```
bfconvert -precompressed /path/to/input /path/to/output
```

For an example of this option being used see below using the CMU-1.svs file from OpenSlide:

```
bfconvert -noflat -precompressed CMU-1.svs CMU-1.dcm
```

Added in version 7.1.0.

See also

Summary of the precompressed tiles feature

-extra-metadata METADATA FILE

Enables the writing of additional supplemental metadata from an external file during the conversion process . Most output formats do not support this option, currently this is implemented for outputting the DICOM format. For DICOM the additional metadata file can be in the format of a .json file

```
bfconvert -extra-metadata /path/to/extra-metadata /path/to/input /path/to/output
```

For an example of this option being used see below:

```
$ cat hierarchy-test.json
{
   "BodyPartExamined": {
     "Value": "BRAIN",
     "VR": "CS",
     "Tag": "(0018,0015)"
     "ResolutionStrategy": "IGNORE"
   },
   "ContributingEquipmentSequence": {
     "VR": "SQ",
     "Tag": "(0018,a001)",
     "Sequence": {
       "Manufacturer": {
         "Value": "PixelMed",
         "VR": "LO",
         "Tag": "(0008,0070)"
       },
       "ContributionDateTime": {
         "Value": "20210710234601.105+0000",
         "VR": "DT",
         "Tag": "(0018,a002)"
       }
     }
      "ResolutionStrategy": "REPLACE"
   }
}
$ bfconvert -extra-metadata hierarchy-test.json test.fake hierarchy-test-json.dcm
```

Added in version 7.1.0.

2.2.4 Validating XML in an OME-TIFF

The XML stored in an OME-TIFF file can be validated using the *command line tools*.

Both the **tiffcomment** and **xmlvalid** commands are used; **tiffcomment** extracts the XML from the file and **xmlvalid** validates the XML and prints any errors to the console.

For example:

```
xmlvalid /path/to/file.ome.tiff
```

and

```
xmlvalid /path/to/file.xml
```

will perform validatation for an ome.xml file and xml file, respectively.

Also:

```
tiffcomment "/path/to/file.ome.tiff" | xmlvalid
```

will perform the extraction and validation all at once.

Typical successful output of **xmlvalid** is:

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

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:

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

If the XML is found to have validation errors, the **tiffcomment** command can be used to overwrite the XML in the OME-TIFF file with corrected XML. The XML can be displayed in an editor window:

```
tiffcomment -edit /path/to/file.ome.tiff
```

or the new XML can be read from a file:

```
tiffcomment -set new-comment.xml /path/to/file.ome.tiff
```

2.2.5 Editing XML in an OME-TIFF

To edit the XML in an OME-TIFF file you can use **tiffcomment**, one of the Bio-Formats tools.

> **Note**
>
> The **tiffcomment** tool requires that the *ImageDescription* tag is present in the TIFF file and will error otherwise.

To use the built in editor run:

```
tiffcomment -edit sample.ome.tif
```

To extract or view the XML run:

```
tiffcomment sample.ome.tif
```

To inject replacement XML into a file run:

```
tiffcomment -set 'newmetadata.xml' sample.ome.tif
```

2.2.6 List formats by domain

Each supported file format has one or more imaging domains associated with it. To print the list of formats associated with each imaging domain:

```
domainlist
```

The command does not accept any arguments. The known image domains are defined by:

- ASTRONOMY_DOMAIN
- EM_DOMAIN
- FLIM_DOMAIN
- GEL_DOMAIN
- GRAPHICS_DOMAIN
- HCS_DOMAIN
- HISTOLOGY_DOMAIN
- LM_DOMAIN
- MEDICAL_DOMAIN
- SEM_DOMAIN
- SPM_DOMAIN
- UNKNOWN_DOMAIN

2.2.7 List supported file formats

A detailed list of supported formats can be displayed using the **formatlist** command.

The default behavior is to print a plain-text list of formats:

```
formatlist
```

-txt

Prints the list of formats as plain-text:

```
formatlist -txt
```

-html

Prints the list of formats as HTML:

```
formatlist -html
```

-xml

Prints the list of formats as XML:

```
formatlist -xml
```

-help

Displays the usage information:

```
formatlist -help
```

2.2.8 Display file in ImageJ

Files can be displayed from the command line in ImageJ. The Bio-Formats importer plugin for ImageJ is used to open the file.

The command takes a single argument:

```
ijview /file/to/open
```

If the input file is not specified, ImageJ will show a file chooser window.

The Bio-Formats import options window will then appear, after which the image(s) will be displayed.

If the *BF_DEVEL* environment variable is set, the `ImageJ jar <jars/ij.jar>` must be included in the classpath.

2.2.9 Format XML data

The **xmlindent** command formats and adds indenting to XML so that it is easier to read. Indenting is currently set to 3 spaces.

If an XML file name is not specified, the XML to indent will be read from standard output. Otherwise, one or more file names can be specified:

```
xmlindent /path/to/xml
xmlindent /path/to/first-xml /path/to/second-xml
```

The formatted XML from each file will be printed in the order in which the files were specified.

By default, extra whitespace may be added to CDATA elements. To preserve the contents of CDATA elements:

```
xmlindent -valid /path/to/xml
```

2.2.10 Create a high-content screen for testing

The **mkfake** command creates a high-content screen for testing. The image data will be meaningless, but it allows testing of screen, plate, and well metadata without having to find appropriately-sized screens from real acquisitions.

If no arguments are specified, **mkfake** prints usage information.

To create a single screen with default plate dimensions:

```
mkfake default-screen.fake
```

This will create a directory that represents one screen with a single plate containing one well, one field, and one acquisition of the plate (see PlateAcquisition).

-plates PLATES

To change the number of plates in the screen:

```
mkfake -plates 3 three-plates.fake
```

-runs RUNS

To change the number of acquisitions for each plate:

```
mkfake -runs 4 four-plate-acquisitions.fake
```

-rows ROWS

To change the number of rows of wells in each plate:

```
mkfake -rows 8 eight-row-plate.fake
```

-columns COLUMNS

To change the number of columns of wells in each plate:

```
mkfake -columns 12 twelve-column-plate.fake
```

-fields FIELDS

To change the number of fields per well:

```
mkfake -fields 2 two-field-plate.fake
```

It is often most useful to use the arguments together to create a realistic screen, for example:

```
mkfake -rows 16 -columns 24 -plates 2 -fields 3 two-384-well-plates.fake
```

-debug DEBUG

As with other command line tools, debugging output can be enabled if necessary:

```
mkfake -debug debug-screen.fake
```

2.2.11 "Precompressed" tiles

See first *an overview of what precompressed tiles are and which formats support them*.

This feature is primarily exposed in the **bfconvert** command, using the *-precompressed* option. When the *-precompressed* option is used, best results are obtained by:

- using the *-noflat* option
- **not** using the *-compression* option (this can work, but may force tile recompression)
- **not** using the *-tilex* or *-tiley* options (this can work, but may force tile recompression)

There are several advantages to converting using the "precompressed" feature for formats that support it:

- faster tile read and write times, as no tile compression/decompression needs to be performed
- no change in compression quality

- very little change in file sizes between input and output data

There are also a few disadvantages:

- tile sizes cannot be changed during conversion
- compression type cannot be changed during conversion
- input and output format must support same compression type and tile size
- the input format must have precompressed tile reading support (see *the list of supported readers*)
- the output format must have precompressed tile writing support (see *the list of supported writers*)
- command-line tools other than **`bfconvert`** do not currently make use of the precompressed tile API

For additional discussion of the "precompressed" tiles feature, see:

- https://forum.image.sc/t/vsi-to-dcm-file-conversion/98249/5
- https://forum.image.sc/t/exporting-a-dicom-stack/99400

See also

Additional reader and writer options

2.3 Using Bio-Formats in MATLAB/Octave

The following pages explain how to install and use Bio-Formats in MATLAB or Octave:

2.3.1 MATLAB

MATLAB is a high-level language and interactive environment that facilitates rapid development of algorithms for performing computationally intensive tasks.

Calling Bio-Formats from MATLAB is fairly straightforward, since MATLAB has built-in interoperability with Java. We have created a toolbox for reading and writing image files. Note the minimum recommended MATLAB version is R2017b.

Note

It is possible to run Bio-Formats 6 on earlier MATLAB versions using a JVM version 8 or greater although using a different JVM than the one shipped with MATLAB can affect other functionalities. Please refer to the MATLAB Answers for more information.

Installation

Download the MATLAB toolbox from the Bio-Formats downloads page. Unzip `bfmatlab.zip` and add the unzipped `bfmatlab` folder to your MATLAB path.

Usage

Please see *Using Bio-Formats in MATLAB* for usage instructions. If you intend to extend the existing .m files, please also see the *developer page* for more information on how to use Bio-Formats in general.

Performance

In our tests (MATLAB R14 vs. java 1.6.0_20), the script executes at approximately half the speed of our *showinf command line tool*, due to overhead from copying arrays.

Troubleshooting

If you encounter an error trying to open JPEG-2000 data in MATLAB but the file will open e.g. in Fiji using Bio-Formats, it may be due to conflicting versions of JAI ImageIO in different JARs. As discussed on the component page, *JAI ImageIO* is no longer maintained and you will likely need to remove the conflicting JAR(s) as a workaround.

Upgrading

To use a newer version of Bio-Formats, overwrite the content of the `bfmatlab` folder with the newer version of the toolbox and restart MATLAB.

Alternative scripts

Several other groups have developed their own MATLAB scripts that use Bio-Formats, including the following:

- https://github.com/pramukta/bf-tools
- imread for multiple life science image file formats

2.3.2 GNU Octave

GNU Octave is a high-level interpreted language, primarily intended for numerical computations. Being an array programming language, it is naturally suited for image processing and handling of N dimensional datasets. Octave is distributed under the terms of the GNU General Public License.

The Octave language is MATLAB compatible so that programs are easily portable. Indeed, the Octave bioformats package is exactly the same as MATLAB's, the only difference being the installation steps.

Requirements

The bioformats package requires Octave version 4.0.0 or later with support for java:

```
$ octave
>> OCTAVE_VERSION
ans = 4.0.0
>> usejava ("jvm")
ans =  1
```

Installation

1. Download bioformats_package.jar and place it somewhere sensible for your system (in Linux, this will probably be */usr/local/share/java* or *~/.local/share/java* for a system-wide or user installation respectively).
2. Add *bioformats_package.jar* to Octave's *static* javaclasspath (see Octave's documentation).
3. Download the Octave package.
4. Start octave and install the package with:

```
>> pkg install path-to-bioformats-octave-version.tar.gz
```

Usage

Usage instructions are the same as MATLAB. The only difference is that you need to explicitly load the package. This is done by running at the Octave prompt:

```
>> pkg load bioformats
```

Upgrading

To use a newer version of Bio-Formats, repeat the install instructions. Do not follow the MATLAB instructions.

2.4 Third-party applications

Several third-party libraries and applications use Bio-Formats to read and write imaging formats. The list below includes the projects registered as Community partners on the image.sc forum which are known to use Bio-Formats for reading imaging file formats. The forum is the primary discussion channel for these software projects and should be used for any questions regarding installation, development or usage:

- BiofilmQ
- BioFormats.NET
- CellProfiler
- Icy
- MIB
- Manager
- Orbit
- OMERO
- PolusAI
- QuPath

CHAPTER

THREE

DEVELOPER DOCUMENTATION

The following sections describe various things that are useful to know when working with Bio-Formats. It is recommended that you obtain the Bio-Formats source by following the directions in the *Source code* section. Referring to the Javadocs as you read over these pages should help, as the notes will make more sense when you see the API.

For a complete list of supported formats, see the Bio-Formats *supported formats table*.

For a few working examples of how to use Bio-Formats, see these Github pages and the bio-formats-examples repository.

3.1 Introduction to Bio-Formats

3.1.1 Overview for developers

From the rest of the Bio-Formats developer documentation one may piece together a correct and useful understanding of what Bio-Formats does and how it does it. This section gives a high-level tour of these technical details, for those new to working on Bio-Formats itself, making it easier to understand how the information from the other sections fits into the big picture.

Terms and concepts

Bio-Formats can read image data from files for many formats, and can write image data to files for some formats. An image may have many two-dimensional "planes" of pixel intensity values. Each pixel on a plane is identified by its x, y values. Planes within an image may be identified by various dimensions including z (third spatial dimension), c (channel, e.g. wavelength) or t (time). Planes may be divided into tiles, which are rectangular subsections of a plane; this is helpful in handling very large planes. A file (or set of related files) on disk may contain multiple images: each image is identified by a unique *series* number.

An image is more than a set of planes: it also has metadata. Bio-Formats distinguishes *core metadata*, such as the x, y, z, c, t dimensions of the image, from format-specific *original metadata*, e.g. information about the microscope and its settings, which is represented as a dictionary of values indexed by unique keys. Metadata apply to the image data as a whole, or separately to specific series within it.

Bio-Formats is able to translate the above metadata into a further form, *OME metadata*. The translation may be partial or incomplete, but remains very useful for allowing the metadata of images from different file formats to be used and compared in a common format defined by the OME data model.

Implementation

Bio-Formats is primarily a Java project. It can be used from MATLAB and there is also a separate C++ implementation (OME Files C++). The source code is available for download and sometimes the user community contributes code back into Bio-Formats by opening a pull request on GitHub. Bio-Formats is built from source with Ant or Maven and some of the Bio-Formats source code is generated from other files during the build process. The resulting JARs corresponding to official Bio-Formats releases are available for download.

Readers and writers for different image file formats are implemented in separate Java classes. Readers for related formats may reflect that relationship in the Java class hierarchy. Simple standalone command-line tools are provided with Bio-Formats, but it is more commonly used as a third-party library by other applications. Various examples show how one may use Bio-Formats in different ways in writing a new application that reads or writes image data. A common pattern is to initialize a reader based on the image data's primary file, then query that reader for the metadata and planes of interest.

The set of readers is easily modified. The readers.txt file lists the readers to try in determining an image file's format, and there are many useful classes and methods among the Bio-Formats Java code to assist in writing new readers and writers.

Deprecated API

Some classes and API methods are marked as deprecated, using the *@Deprecated* annotation. These classes and/or methods may be removed in a future release, and so their use should be avoided where possible. If not possible, please report an issue outlining the need to keep a class or method marked as deprecated.

3.1.2 Obtaining and building Bio-Formats

> **Note**
>
> Bio-Formats requires Java 8 or above

Source code

The source code for this Bio-Formats release is available from the downloads site. This release and the latest Bio-Formats source code are also available from the Git repository. This may be accessed using the repository path:

```
git@github.com:openmicroscopy/bioformats.git
```

More information about Git and client downloads are available from the Git project website. You can also browse the Bio-Formats source on GitHub

> **Note**
>
> Windows users must set git to use `core.autocrlf=input` to ensure that Bio-Formats uses LF rather than CRLF line endings, otherwise the build will fail (Genshi can't process code templates with CRLF line endings, leading to broken sources being generated). This can be set globally in the registry when installing **`msysgit`** or by editing `etc/gitconfig` in the git installation directory. Annoyingly, these settings appear to override per-user and per-repository configuration values, requiring these to be set globally.

Lastly, you can browse the Bio-Formats Javadocs online, or generate them yourself using the "docs" Ant target.

Source code structure

The Bio-Formats code is divided into several projects. Core components are located in subfolders of the components folder, with some components further classified into components/forks, depending on the nature of the project. See the *Component overview* for more information, including associated build targets for each component.

Each project has a corresponding Maven POM file, which can be used to work with the project in your favorite IDE, or from the command line, once you have cloned the source.

Building from source

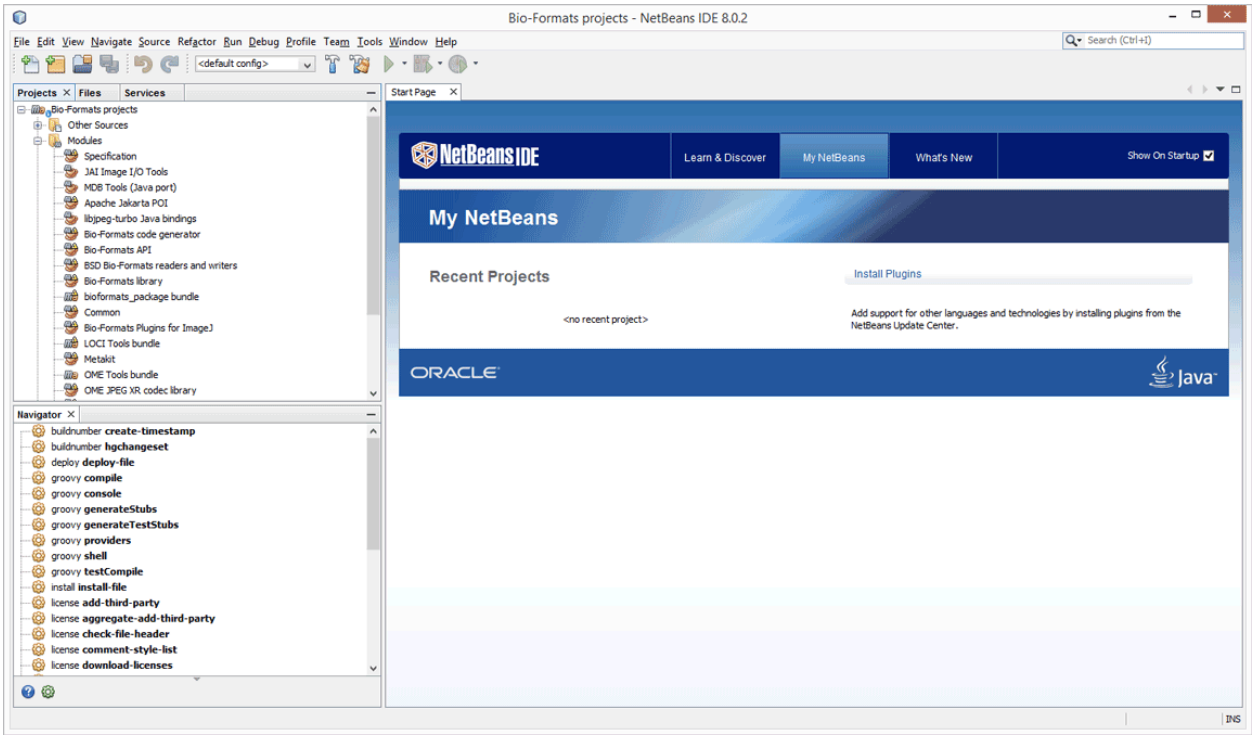
Instructions for several popular options follow. In all cases, make sure that the prerequisites are installed before you begin.

If you are interested in working on the Bio-Formats source code itself, you can load it into your favorite IDE, or develop with your favorite text editor.

NetBeans

NetBeans comes with Maven support built in. To import the Bio-Formats source, perform the following steps:

1. Select *File* → *Open Project* from the menu - choose the top-level path to bioformats.git and click *Open Project*
2. In the 'Projects' tab on the left-hand side, expand the 'Bio-Formats projects' entry - you should now have a series of folders including 'Other Sources', 'Modules' and 'Dependencies.
3. Expand the 'Modules' folder to give a list of components and then double-click the desired project(s) to work with them.



Alternately, you can clone the source directly from NetBeans into a project by selecting *Team* → *Git* → *Clone Other...* from the menu.

Eclipse

Eclipse uses the "Maven Integration for Eclipse" (m2e) plugin to work with Maven projects. It is more flexible than Eclipse's built-in project management because m2e transparently converts between project dependencies and JAR dependencies (stored in the Maven repository in `~/.m2/repository`) on the build path, depending on which projects are currently open.

We recommend using Eclipse 4.3 (Kepler) or later, specifically - "Eclipse IDE for Java developers". It comes with m2e installed (http://eclipse.org/downloads/compare.php?release=kepler).

You can import the Bio-Formats source by choosing *File* → *Import* → *Existing Maven Projects* from the menu and browsing to the top-level folder of your Bio-Formats working copy. Alternatively, run the Eclipse Maven target with **`mvn eclipse:eclipse`** to create the Eclipse project files, then use *File* → *Import* → *Existing Maven Projects*.

IntelliJ IDEA

IntelliJ IDEA can build Bio-Formats using either Maven or Ant. Go to *File* → *Open...* and browse to the top-level folder of your Bio-Formats working copy. Select *Build* → *Build Project* or select the *Maven* tab on the right of the screen.

Command line

If you prefer developing code with a text editor such as vim or emacs, you can use the Ant or Maven command line tools to compile Bio-Formats. The Bio-Formats source tree provides parallel build systems for both Ant and Maven, so you can use either one to build the code.

For a list of Ant targets, run:

```
ant -p
```

In general, `ant jars` or `ant tools` is the correct command.

When using Maven, Bio-Formats is configured to run the "install" target by default, so all JARs will be copied into your local Maven repository in `~/.m2/repository`. Simply run:

```
mvn
```

With either Ant or Maven, you can use similar commands in any subproject folder to build just that component.

3.1.3 Component overview

The Bio-Formats code repository is divided up into separate components.

The Ant targets to build each component from the repository root are noted in the component descriptions below. Unless otherwise noted, each component can also be built with Maven by running **`mvn`** in the component's subdirectory. The Maven module name for each component (as it is shown in most IDEs) is also noted in parenthesis.

Core components

The most commonly used and actively modified components.

- *formats-api*
- *formats-bsd*
- *formats-gpl*

Internal testing components

These components are used heavily during continuous integration testing, but are less relevant for active development work.

- *test-suite*

Forks of existing projects

- *jai*
- *turbojpeg*

All components

bio-formats-plugins (Bio-Formats Plugins for ImageJ):

Ant: jar-bio-formats-plugins

Everything pertaining to the Bio-Formats plugins for ImageJ lives in this component. Note that when built, this component produces `bio-formats_plugins.jar` (instead of `bio-formats-plugins.jar`) to be in keeping with ImageJ plugin naming conventions.

bio-formats-tools (Bio-Formats command line tools):

Ant: jar-bio-formats-tools

The classes that implement the **`showinf`**, **`bfconvert`**, and **`mkfake`** *command line tools* are contained in this component. Note that this is built with the **`jar-bio-formats-tools`** Ant target, and not the **`tools`** target (which is the Ant equivalent of *bundles*).

bundles (bioformats_package bundle, LOCI Tools bundle):

Ant: tools

This is only needed by the Maven build system, and is used to aggregate all of the individual .jar files into `bioformats_package.jar`. There should not be any code here, just build system files.

OME JAI (deprecated):

This is a fork of JAI ImageIO. JAI ImageIO is no longer maintained; the most active fork is jai-imageio-core on GitHub. JAI provides support for decoding YCbCr JPEG-2000 data. This is primarily needed for reading images from histology/pathology formats in *formats-gpl*. There are no dependencies on other components.

The status of this component means that you may encounter errors due to conflicting JARs e.g. between Bio-Formats and other toolboxes within Fiji or MATLAB, especially when trying to open JPEG-2000 data. In this case, you will need to remove the conflicting JAR(s) as a workaround.

forks/turbojpeg (libjpeg-turbo Java bindings):

Ant: jar-turbojpeg

This is a fork of libjpeg-turbo. There are not any real code changes, but having this as a separate component allows us to package the libjpeg-turbo Java API together with all of the required binaries into a single .jar file using native-lib-loader. There are no dependencies on other components.

formats-api (Bio-Formats API):

Ant: jar-formats-api

This defines all of the high level interfaces and abstract classes for reading and writing files. There are no file format readers or writers actually implemented in this component, but it does contain the majority of the API that defines Bio-Formats. *formats-bsd* and *formats-gpl* implement this API to provide file format readers and writers. *ome-common* and *ome-xml* are both required as part of the interface definitions.

formats-bsd (BSD Bio-Formats readers and writers):

Ant: jar-formats-bsd

This contains readers and writers for formats which have a publicly available specification, e.g. TIFF. Everything in the component is BSD-licensed.

formats-gpl (Bio-Formats library):

Ant: jar-formats-gpl

The majority of the file format readers and some file format writers are contained in this component. Everything in the component is GPL-licensed (in contrast with *formats-bsd*). Most file formats represented in this component do not have a publicly available specification.

test-suite (Bio-Formats testing framework):

Ant: jar-tests

All tests that operate on files from our data repository (i.e. integration tests) are included in this component. These tests are primarily run by the continuous integration jobs, and verify that there are no regressions in reading images or metadata.

External components

The following have been decoupled from the Bio-Formats code repository and are now available as separate build dependencies:

- *Bio-Formats examples*
- *Bio-Formats documentation*
- *Metakit*
- *OME Common*
- *OME Codecs*
- *OME MDB Tools (Java)*
- *OME Apache Jakarta POI*
- *JXRlib*

Decoupled OME data model components:

- *OME-XML*
- *Specification*

Bio-Formats examples

Usage examples for Bio-Formats with Maven and Gradle.

Bio-Formats documentation

The Sphinx source repository for this manual.

OME Common (Java / C++):

Provides I/O classes that unify reading from files on disk, streams or files in memory, compressed streams, and non-file URLs. The primary entry points are Location, RandomAccessInputStream (for reading), and RandomAccessOutputStream (for writing).

In addition to I/O, there are several classes to assist in working with XML (XMLTools), date/timestamps (DateTools), logging configuration (DebugTools), and byte arithmetic (DataTools).

OME Codecs:

Provides classes for encoding and decoding compressed data for a variety of compression formats. *ome-common* is a required dependency for I/O and service loading.

OME MDB Tools (Java port):

This is a fork of the mdbtools-java project. There are numerous bug fixes, as well as changes to reduce the memory required for large files. There are no dependencies on other components.

OME Apache Jakarta POI:

This is a fork of Apache POI, which allows reading of Microsoft OLE document files. We have made substantial changes to support files larger than 2GB and reduce the amount of memory required to open a file. I/O is also handled by classes from *ome-common*, which allows OLE files to be read from memory.

Metakit Java library:

Java implementation of the Metakit database specification. This uses classes from *ome-common* and is used by *formats-gpl*, but is otherwise independent of the main Bio-Formats API.

OME-XML Java library:

This component contains classes that represent the OME-XML schema. Some classes are committed to the Git repository, but the majority are generated at build time by using XSD-FU to parse the *OME-XML schema files*. Classes from this component are used by Bio-Formats to read and write OME-XML, but they can also be used independently.

Model specification:

All released and in-progress OME-XML schema files are contained in this component. The specification component is also the location of all XSLT stylesheets for converting between OME-XML schema versions, as well as example OME-XML files in each of the released schema versions.

Stubs:

MIPAV stubs.

This component provides empty classes that mirror third-party dependencies which are required at compile time but cannot be included in the build system (usually due to licensing issues). The build succeeds since required class names are present with the correct method signatures; the end user is then expected to replace the stub .jar files at runtime.

JXRlib:

This component contains the Java bindings for jxrlib, an open source implementation of the JPEG-XR image format standard.

3.1.4 Reading files

Basic file reading

Bio-Formats provides several methods for retrieving data from files in an arbitrary (supported) format. These methods fall into three categories: raw pixels, core metadata, and format-specific metadata. All methods described here are present and documented in loci.formats.IFormatReader. In general, it is recommended that you read files using an instance of loci.formats.ImageReader. While it is possible to work with readers for a specific format, ImageReader contains additional logic to automatically detect the format of a file and delegate subsequent calls to the appropriate reader.

Prior to retrieving pixels or metadata, it is necessary to call setId(java.lang.String) on the reader instance, passing in the name of the file to read. Some formats allow multiple series (5D image stacks) per file; in this case you may wish to call setSeries(int) to change which series is being read.

Raw pixels are always retrieved one plane at a time. Planes are returned as raw byte arrays, using one of the openBytes methods.

Core metadata is the general term for anything that might be needed to work with the planes in a file. A list of core metadata fields is given in the table below together with the appropriate accessor method:

Core metadata field	API method
image width	getSizeX()
image height	getSizeY()
number of series per file	getSeriesCount()
total number of images per series	getImageCount()
number of slices in the current series	getSizeZ()
number of timepoints in the current series	getSizeT()
number of actual channels in the current series	getSizeC()
number of channels per image	getRGBChannelCount()
the ordering of the images within the current series	getDimensionOrder()
whether each image is RGB	isRGB()
whether the pixel bytes are in little-endian order	isLittleEndian()
whether the channels in an image are interleaved	isInterleaved()
the type of pixel data in this file	getPixelType()

All file formats are guaranteed to accurately report core metadata.

Bio-Formats also converts and stores additional information which can be stored and retrieved from the OME-XML Metadata. These fields can be accessed in a similar way to the core metadata above. An example of such values would be the physical size of dimensions X, Y and Z. The accessor methods for these properties return a Length object which contains both the value and unit of the dimension. These lengths can also be converted to other units using value(ome.units.unit.Unit) An example of reading and converting these physical sizes values can be found in `ReadPhysicalSize.java`.

Format-specific metadata refers to any other data specified in the file - this includes acquisition and hardware parameters, among other things. This data is stored internally in a **java.util.Hashtable**, and can be accessed in one of two ways: individual values can be retrieved by calling getMetadataValue(java.lang.String), which gets the value of the specified key. Note that the keys in this Hashtable are different for each format, hence the name "format-specific metadata".

See *Bio-Formats metadata processing* for more information on the metadata capabilities that Bio-Formats provides.

See also

IFormatReader
: Source code of the `loci.formats.IFormatReader` interface

`OrthogonalReader.java`
: Example of reading XZ and YZ image planes from a file

File reading extras

The previous section described how to read pixels as they are stored in the file. However, the native format is not necessarily convenient, so Bio-Formats provides a few extras to make file reading more flexible.

- The loci.formats.ReaderWrapper API that implements `loci.formats.IFormatReader` allows to define "wrapper" readers that take a reader in the constructor, and manipulate the results somehow, for convenience. Using them is similar to the java.io InputStream/OutputStream model: just layer whichever functionality you need by nesting the wrappers.

  The table below summarizes a few wrapper readers of interest:

Wrapper reader	Functionality
loci.formats.BufferedImag	Allows pixel data to be returned as BufferedImages instead of raw byte arrays
loci.formats.FileStitcher	Uses advanced pattern matching heuristics to group files that belong to the same dataset
loci.formats.ChannelSepa	Makes sure that all planes are grayscale - RGB images are split into 3 separate grayscale images
loci.formats.ChannelMerg	Merges grayscale images to RGB if the number of channels is greater than 1
loci.formats.ChannelFiller	Converts indexed color images to RGB images
loci.formats.MinMaxCalc	Provides an API for retrieving the minimum and maximum pixel values for each channel
loci.formats.DimensionSw	Provides an API for changing the dimension order of a file
loci.formats.Memoizer	Caches the state of the reader into a memoization file

- loci.formats.ImageTools and loci.formats.gui.AWTImageTools provide a number of methods for manipulating BufferedImages and primitive type arrays. In particular, there are methods to split and merge channels in a BufferedImage/array, as well as converting to a specific data type (e.g. convert short data to byte data).

Troubleshooting

- Importing multi-file formats (Leica LEI, PerkinElmer, FV1000 OIF, ICS, and Prairie TIFF, to name a few) can fail if any of the files are renamed. There are "best guess" heuristics in these readers, but they are not guaranteed to work in general. So please do not rename files in these formats.
- If you are working on a Macintosh, make sure that the data and resource forks of your image files are stored together. Bio-Formats does not handle separated forks (the native QuickTime reader tries, but usually fails).
- Bio-Formats file readers are not thread-safe. If files are read within a parallelized environment, a new reader must be fully initialized in each parallel session. See *Improving reading performance* about ways to improve file reading performance in multi-threaded mode.

3.1.5 Writing files

The loci.formats.IFormatWriter API is very similar to the reader API, in that files are written one plane at time (rather than all at once).

The file formats which can be written using Bio-Formats are marked in the *supported formats table* with a green tick in the 'export' column. These include, but are not limited to:

- TIFF (uncompressed, LZW, JPEG, or JPEG-2000)
- OME-TIFF (uncompressed, LZW, JPEG, or JPEG-2000)
- JPEG
- PNG
- AVI (uncompressed)
- QuickTime (uncompressed)
- Encapsulated PostScript (EPS)
- OME-XML (not recommended)

All writers allow the output file to be changed before the last plane has been written. This allows you to write to any number of output files using the same writer and output settings (compression, frames per second, etc.), and is especially useful for formats that do not support multiple images per file.

See also

IFormatWriter
Source code of the `loci.formats.IFormatWriter` interface

loci.formats.tools.ImageConverter
Source code of the `loci.formats.tools.ImageConverter` class

Further details on exporting raw pixel data to OME-TIFF files
Examples of OME-TIFF writing

3.2 Using Bio-Formats as a Java library

3.2.1 Using Bio-Formats as a Java library

Bio-Formats as a Maven, Gradle or Ivy dependency

All released `.jar` artifacts are published to and can be obtained from the OME Artifactory. The "Client Settings" section of the Artifactory main page provides example code snippets for inclusion into your Gradle, Maven or Ivy project, which will enable the use of this repository.

Changed in version 6.12.0.

As of Bio-Formats 6.12.0 consumers will have to directly declare a dependency for their chosen SLF4J logging framework. See *Logging* for further details.

Examples of getting started with Bio-Formats using Maven or Gradle are given in the https://github.com/ome/bio-formats-examples repository. OMERO uses Ivy to manage its Java dependencies including Bio-Formats.

Note

In order to retrieve the NetCDF dependency of Bio-Formats, it is necessary to configure the Unidata releases repository in addition to Maven Central and the OME artifactory in your https://github.com/ome/bio-formats-examples/blob/master/pom.xml, https://github.com/ome/bio-formats-examples/blob/master/build.gradle or `ivy.xml` file.

Bio-Formats as a Java library

Alternatively Bio-Formats can be used by including its component jar files. You can download formats-gpl.jar to use it as a library. Just add `formats-gpl.jar` to your `CLASSPATH` or build path. You will also need `ome-common.jar` for common I/O functions, `ome-xml.jar` for metadata standardization, and a SLF4J binding for *Logging*.

Dependencies

The complete list of third-party dependencies for *formats-gpl.jar* is as follows:

Package	Maven name	License
JHDF5 v19.04.1	cisd:jhdf5:19.04.1	Apache License v2.0
cisd base v18.09.0	cisd:base:18.09.0	Apache License v2.0
commons-io v2.6	commons-io:commons-io:2.6	Apache License v2.0
commons-lang3 v3.18.0	org.apache.commons:commons-lang3:3.18.0	Apache License v2.0
XMP Library for Java v6.1.11	com.adobe.xmp:xmpcore:6.1.11	BSD 3-Clause

continues on next page

Table 1 – continued from previous page

Package	Maven name	License
JCommander v1.27	com.beust:jcommander:1.27	Apache License v2.0
metadata-extractor v2.18.0	com.drewnoakes:metadata-extractor:2.18.0	Apache License v2.0
aircompressor v2.0.3	io.airlift:aircompressor:2.0.3	Apache License v2.0
Kryo v5.4.0	com.esotericsoftware.kryo:kryo:5.4.0	BSD 3-Clause
reflectasm v1.11.9	com.esotericsoftware:reflectasm:1.11.	BSD 3-Clause
MinLog v1.3.1	com.esotericsoftware.minlog:minlog:	BSD 3-Clause
Guava v32.0.1	com.google.guava:guava:32.0.1-jre	Apache License v2.0
JGoodies Common v1.7.0	com.jgoodies:jgoodies-common:1.7.0	BSD License
JGoodies Forms v1.7.2	com.jgoodies:jgoodies-forms:1.7.2	BSD License
Commons Logging v1.2	commons-logging:commons-logging:1.2	Apache License v2.0
NetCDF-Java Library v5.6.0	edu.ucar:cdm-core:5.6.0	BSD 3-Clause
Joda time v2.12.7	joda-time:joda-time:2.12.7	Apache License v2.0
JUnit v4.10	junit:junit:4.10	Common Public License v1.0
BeanShell v2.0b4	org.beanshell:bsh:2.0b4	Sun Public License / LGPL
Hamcrest Core v1.1	org.hamcrest:hamcrest-core:1.1	BSD 3-Clause
Objenesis v3.3	org.objenesis:objenesis:3.3	Apache License v2.0
Perf4J v0.9.16	org.perf4j:perf4j:0.9.16	Apache License v2.0
Native Library Loader v2.4.0	org.scijava:native-lib-loader:2.4.0	BSD License
SLF4J API v2.0.9	org.slf4j:slf4j-api:2.0.9	MIT License
TestNG v6.8	org.testng:testng:6.8	Apache License v2.0
SnakeYAML v2.0	org.yaml:snakeyaml:2.0	Apache License v2.0
Xalan Java Serializer v2.7.2	xalan:serializer:2.7.2	Apache License v2.0
Xalan Java v2.7.3	xalan:xalan:2.7.3	Apache License v2.0
Xerces2 Java Parser v2.12.2	xerces:xercesImpl:2.12.2	Apache License v2.0
XML Commons External Components XML APIs v1.4.01	xml-apis:xml-apis:1.4.01	Apache License v2.0
minio v5.0.2	io.minio:minio:5.0.2	Apache License v2.0
http-client-xml v1.20.0	com.google.http-client:google-http-client-xml:1.20.0	Apache License v2.0
http-client v1.20.0	com.google.http-client:google-http-client:1.20.0	Apache License v2.0
xpp3 v1.1.4c	xpp3:xpp3:1.1.4c	https://github.com/aslom/xpp3/blob/master/LICENSE.txt
okhttp3 v3.7.0	com.squareup.okhttp3:okhttp:3.7.0	Apache License v2.0
okio v1.12.0	com.squareup.okio:okio:1.12.0	Apache License v2.0
json v20231013	org.json:json:20231013	Public Domain
sqlite v3.49.1.0	org.xerial:sqlite-jdbc:3.49.1.0	Apache License v2.0
failureaccess v1.0.1	com.google.guava:failureaccess:1.0.1	Apache License v2.0
listenablefuture v9999.0	com.google.guava:listenablefuture:99 empty-to-avoid-conflict-with-guava	Apache License v2.0
jsr305 v3.0.2	com.google.code.findbugs:jsr305:3.0.	BSD 3-Clause
checker-qual v3.33.0	org.checkerframework:checker-qual:3.33.0	GPL v2
errorprone v2.18.0	com.google.errorprone:error_prone_a	Apache License v2.0
j2objc-annotations v2.8	com.google.j2objc:j2objc-annotations:2.8	Apache License v2.0
httpservices v5.6.0	edu.ucar:httpservices:5.6.0	BSD 3-Clause

continues on next page

Table 1 – continued from previous page

Package	Maven name	License
httpclient v4.5.13	org.apache.httpcomponents:httpclient	Apache License v2.0
commons-codec v1.11	commons-codec:commons-codec:1.11	Apache License v2.0
commons-math v1.2	commons-math:commons-math:1.2	Apache License v2.0
httpmime v4.5.13	org.apache.httpcomponents:httpmime	Apache License v2.0
jackson-core v2.14.2	com.fasterxml.jackson.core:jackson-core:2.14.2	Apache License v2.0
jackson-annotations v2.14.2	com.fasterxml.jackson.core:jackson-annotations:2.14.2	Apache License v2.0
jackson-databind v2.14.2	com.fasterxml.jackson.core:jackson-databind:2.14.2	Apache License v2.0
re2j v1.3	com.google.re2j:re2j:1.3	Go License

The below are optional dependencies :

Package	Maven name	License
Logback Classic v1.3.16	ch.qos.logback:logback-classic:1.3.16	Eclipse Public License v1.0
Logback Core v1.3.16	ch.qos.logback:logback-core:1.3.16	Eclipse Public License v1.0

As described in *Versioning policy*, the minor version number of a Bio-Formats release will always be increased if the version of a non-OME/external dependency is bumped.

Dependency lists for any component can be generated by checking out the code as described in *Source code* and then running:

```
cd components/$COMPONENT_NAME
mvn dependency:tree
```

Examples of usage

File reading and performance:

MultiFileExample - Simple example of how to open multiple files simultaneously.

ParallelRead - Reads all files in given directory in parallel, using a separate thread for each.

ReadWriteInMemory - Tests the Bio-Formats I/O logic to and from byte arrays in memory.

OrthogonalReader - Reads image data in XZ and YZ order.

File writing:

MinimumWriter - A command line utility demonstrating the minimum amount of metadata needed to write a file.

FileExport - Write a file in any supported output format.

TiledExport - Shows how to convert a file one tile at a time, instead of one plane at a time (needed for very large images).

FileExportSPW - Write a file with plate (OME SPW) metadata.

File compression:

makeLZW - Converts the given image file to an LZW-compressed TIFF.

Metadata extract/print:

GetPhysicalMetadata - Uses Bio-Formats to extract some basic standardized (format-independent) metadata.

ReadPhysicalSize - Reads physical size information uses the units API to display in micrometers.

ImageInfo - A more involved command line utility for thoroughly reading an input file, printing some information about it, and displaying the pixels onscreen using the Bio-Formats viewer.

PrintTimestamps - A command line example demonstrating how to extract timestamps from a file.

PrintLensNA - Uses Bio-Formats to extract lens numerical aperture in a format-independent manner from a dataset.

PrintROIs - A simple example of how to retrieve ROI data parsed from a file.

SubResolutionExample - Demonstration of the sub-resolution API.

Metadata add/edit:

EditImageName - Edits the given file's image name (but does not save back to disk).

EditTiffComment - Allows raw user TIFF comment editing for the given TIFF files.

writeMapAnnotations - Example method to write MapAnnotations to the ome-xml.

CommentSurgery - Edits a TIFF ImageDescription comment, particularly the OME-XML comment found in OME-TIFF files.

Image converters:

ImageConverter - A simple command line tool for converting between formats.

FileConvert - Converts a file in any supported format to any supported output format.

ConvertToOmeTiff - Converts the given files to OME-TIFF format.

WritePreCompressedPlanes - Writes the pixels from a set of JPEG files to a single TIFF. The pixel data is used as-is, so no decompression or re-compression is performed.

GeneratePyramidResolutions - Convert a file containing a single large image to a pyramid OME-TIFF.

TiledReaderWriter - Convert a file to OME-TIFF one tile at a time.

OverlappedTiledWriter - Convert a file to OME-TIFF one tile at a time, when the image size is not a multiple of the tile size.

SimpleTiledWriter - Convert a file to OME-TIFF using automatic tiling.

ImageJ plugins:

Simple_Read - A simple ImageJ plugin demonstrating how to use Bio-Formats to read files into ImageJ (see *ImageJ overview*).

Read_Image - An ImageJ plugin that uses Bio-Formats to build up an image stack, reading image planes one by one (see *ImageJ overview*).

Mass_Importer - A simple plugin for ImageJ that demonstrates how to open all image files in a directory using Bio-Formats, grouping files with similar names to avoiding opening the same dataset more than once (see *ImageJ overview*).

Image processing utilities:

SewTiffs - Stitches the first plane from a collection of TIFFs into a single file.

SumPlanes - Sums together the image planes from the given file,and saves the result to a 16-bit TIFF.

A Note on Java Web Start (bioformats_package.jar vs. formats-gpl.jar)

To use Bio-Formats with your Java Web Start application, we recommend using **formats-gpl.jar** rather than **bioformats_package.jar**—the latter is merely a bundle of **formats-gpl.jar** plus all its optional dependencies.

The **bioformats_package.jar** bundle is intended as a convenience (e.g. to simplify installation as an ImageJ plugin), but is by no means the only solution for developers. We recommend using **formats-gpl.jar** as a separate entity depending on your needs as a developer.

The bundle is quite large because we have added support for several formats that need large helper libraries (e.g. Imaris' HDF-based format). However, these additional libraries are optional; Bio-Formats has been coded using reflection so that it can both compile and run without them.

When deploying a JNLP-based application, using **bioformats_package.jar** directly is not the best approach, since every time Bio-Formats is updated, the server would need to feed another 15+ MB JAR file to the client. Rather, Web Start is a case where you should keep the JARs separate, since JNLP was designed to make management of JAR dependencies trivial for the end user. By keeping **formats-gpl.jar** and the optional dependencies separate, only a <1 MB JAR needs to be updated when **formats-gpl.jar** changes.

As a developer, you have the option of packaging **formats-gpl.jar** with as many or as few optional libraries as you wish, to cut down on file size as needed. You are free to make whatever kind of "stripped down" version you require. You could even build a custom **formats-gpl.jar** that excludes certain classes, if you like.

3.2.2 Units of measurement

Since Bio-Formats 5.1 and the adoption of the 2015-01 OME Data Model, the data model and the corresponding Bio-Formats model and metadata APIs have added support for units of measurement. Previously, the units for various properties such as the physical size of an image, stage position, confocal pinhole size, light wavelengths etc. were fixed in the model. This was however somewhat inflexible, and not appropriate for imaging modalities at widely different scales. The solution to this was to add a unit of measurement to each of these properties. The image size, for example, was previously specified to be stored in micrometers but may now be specified in any SI length unit of choice, or one of the supported non-SI length units. This permits the preservation of the unit used by a proprietary file format or used at acquisition time, for example nanometers, millimeters, meters, or inches or thousandths of an inch could be used instead.

At the OME-XML level, the properties continue to use the old attribute names. They are supplemented by an additional attribute with a `Unit` suffix, for example the `PhysicalSizeX` attribute and its companion `PhysicalSizeXUnit` attribute.

At the API level, two classes are used:

`Unit<T>`
: represents a unit system for a given dimension such as length, pressure or time.

`Quantity`
: represents a value and unit in a given unit system; this is subclassed for each of the supported dimensions such as `Length`, `Pressure` etc. For example the `Length` class could represent the value and unit of 5.3 µm and the `Pressure` class 956 mbar.

All of the model and metadata APIs pass `Quantity` objects in place of raw numerical values. Updating your code will require replacing the use of raw values with quantities. Where your code needs to deal with the quantity in a specific unit, for example µm, you will need to perform an explicit unit conversion to transform the value to the required unit.

The three situations you will need to deal with are:

- getting a quantity from a `get` method in the API
- converting a quantity to a desired unit
- setting a quantity with a `set` method in the API (possibly also requiring the creation of a quantity)

Examples of how to use units and quantities for these purposes are shown in the sections *Reading files* (`ReadPhysicalSize` example which uses `getPixelsPhysicalSize` and also demonstrates unit conversion) and *Further details on exporting raw pixel data to OME-TIFF files* (`setPixelsPhysicalSize`).

3.2.3 Exporting files using Bio-Formats

This guide pertains to version 4.2 and later.

Basic conversion

The first thing we need to do is set up a reader:

```
// create a reader that will automatically handle any supported format
IFormatReader reader = new ImageReader();
// tell the reader where to store the metadata from the dataset
MetadataStore metadata;

try {
  ServiceFactory factory = new ServiceFactory();
  OMEXMLService service = factory.getInstance(OMEXMLService.class);
  metadata = service.createOMEXMLMetadata();
}
catch (DependencyException exc) {
  throw new FormatException("Could not create OME-XML store.", exc);
}
catch (ServiceException exc) {
  throw new FormatException("Could not create OME-XML store.", exc);
}

reader.setMetadataStore(metadata);
// initialize the dataset
reader.setId("/path/to/file");
```

Now, we set up our writer:

```
// create a writer that will automatically handle any supported output format
IFormatWriter writer = new ImageWriter();
// give the writer a MetadataRetrieve object, which encapsulates all of the
// dimension information for the dataset (among many other things)
OMEXMLService service = factory.getInstance(OMEXMLService.class);
writer.setMetadataRetrieve(service.asRetrieve(reader.getMetadataStore()));
// initialize the writer
writer.setId("/path/to/output/file");
```

Note that the extension of the file name passed to 'writer.setId(...)' determines the file format of the exported file.

Now that everything is set up, we can start writing planes:

```
for (int series=0; series<reader.getSeriesCount(); series++) {
  reader.setSeries(series);
```

(continues on next page)

(continued from previous page)

```
  writer.setSeries(series);

  for (int image=0; image<reader.getImageCount(); image++) {
    writer.saveBytes(image, reader.openBytes(image));
  }
}
```

Finally, make sure to close both the reader and the writer. Failure to do so can cause:

- file handle leaks
- memory leaks
- truncated output files

Fortunately, closing the files is very easy:

```
reader.close();
writer.close();
```

Converting to multiple files

The recommended method of converting to multiple files is to use a single IFormatWriter, like so:

```
// you should have set up a reader as in the first example
ImageWriter writer = new ImageWriter();
OMEXMLService service = factory.getInstance(OMEXMLService.class);
writer.setMetadataRetrieve(service.asRetrieve(reader.getMetadataStore()));
// replace this with your own filename definitions
// in this example, we're going to write half of the planes to one file
// and half of the planes to another file
String[] outputFiles =
  new String[] {"/path/to/file/1.tiff", "/path/to/file/2.tiff"};
writer.setId(outputFiles[0]);

int planesPerFile = reader.getImageCount() / outputFiles.length;
for (int file=0; file<outputFiles.length; file++) {
  writer.changeOutputFile(outputFiles[file]);
  for (int image=0; image<planesPerFile; image++) {
    int index = file * planesPerFile + image;
    writer.saveBytes(image, reader.openBytes(index));
  }
}

reader.close();
writer.close();
```

The advantage here is that the relationship between the files is preserved when converting to formats that support multi-file datasets internally (namely OME-TIFF). If you are only converting to graphics formats (e.g. JPEG, AVI, MOV), then you could also use a separate IFormatWriter for each file, like this:

```
OMEXMLService service = factory.getInstance(OMEXMLService.class);
// again, you should have set up a reader already
String[] outputFiles = new String[] {"/path/to/file/1.avi", "/path/to/file/2.avi"};
```

(continues on next page)

(continued from previous page)

```
int planesPerFile = reader.getImageCount() / outputFiles.length;
for (int file=0; file<outputFiles.length; file++) {
  ImageWriter writer = new ImageWriter();
  writer.setMetadataRetrieve(service.asRetrieve(reader.getMetadataStore()));
  writer.setId(outputFiles[file]);
  for (int image=0; image<planesPerFile; image++) {
    int index = file * planesPerFile + image;
    writer.saveBytes(image, reader.openBytes(index));
  }
  writer.close();
}
```

3.2.4 Further details on exporting raw pixel data to OME-TIFF files

This document explains how to export pixel data to OME-TIFF using Bio-Formats version 4.2 and later.

Working example code is provided in `FileExport.java` - code from that class is referenced here in part. You will need to have `bioformats_package.jar` in your Java `CLASSPATH` in order to compile `FileExport.java`.

The first thing that must happen is we must create the object that stores OME-XML metadata. This is done as follows:

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

The 'omexml' object can now be used in our code to store OME-XML metadata, and by the file format writer to retrieve OME-XML metadata.

Now that we have somewhere to put metadata, we need to populate as much metadata as we can. The minimum amount of metadata required is:

- endianness of the pixel data
- the order in which dimensions are stored
- the bit depth of the pixel data
- the number of channels
- the number of timepoints
- the number of Z sections
- the width (in pixels) of an image
- the height (in pixels) of an image
- the number of samples per channel (3 for RGB images, 1 otherwise)

We populate that metadata as follows:

```
omexml.setImageID("Image:0", 0);
omexml.setPixelsID("Pixels:0", 0);

// specify that the pixel data is stored in big-endian order
// replace 'TRUE' with 'FALSE' to specify little-endian order
omexml.setPixelsBinDataBigEndian(Boolean.TRUE, 0, 0);
```

(continues on next page)

(continued from previous page)

```
omexml.setPixelsDimensionOrder(DimensionOrder.XYCZT, 0);
omexml.setPixelsType(PixelType.UINT16, 0);
omexml.setPixelsSizeX(new PositiveInteger(width), 0);
omexml.setPixelsSizeY(new PositiveInteger(height), 0);
omexml.setPixelsSizeZ(new PositiveInteger(zSectionCount), 0);
omexml.setPixelsSizeC(new PositiveInteger(channelCount *
samplesPerChannel), 0);
omexml.setPixelsSizeT(new PositiveInteger(timepointCount), 0);

for (int channel=0; channel<channelCount; channel++) {
  omexml.setChannelID("Channel:0:" + channel, 0, channel);
  omexml.setChannelSamplesPerPixel(new PositiveInteger(samplesPerChannel),
    0, channel);
}

Unit<Length> unit = UNITS.MICROMETER;
Length physicalSizeX = new Length(1.0, unit);
Length physicalSizeY = new Length(1.5, unit);
Length physicalSizeZ = new Length(2, unit);
omexml.setPixelsPhysicalSizeX(physicalSizeX, 0);
omexml.setPixelsPhysicalSizeY(physicalSizeY, 0);
omexml.setPixelsPhysicalSizeZ(physicalSizeZ, 0);
```

There is much more metadata that can be stored; please see the Javadoc for loci.formats.meta.MetadataStore for a complete list.

Now that we have defined all of the metadata, we need to create a file writer:

```
ImageWriter writer = new ImageWriter();
```

Now we must associate the 'omexml' object with the file writer:

```
writer.setMetadataRetrieve(omexml);
```

The writer now knows to retrieve any metadata that it needs from 'omexml'.

We now tell the writer which file it should write to:

```
writer.setId("output-file.ome.tiff");
```

It is critical that the file name given to the writer ends with ".ome.tiff" or ".ome.tif", as it is the file name extension that determines which format will be written.

Now that everything is set up, we can save the image data. This is done plane by plane, and we assume that the pixel data is stored in a 2D byte array 'pixelData':

```
  int sizeC = omexml.getPixelsSizeC(0).getValue();
  int sizeZ = omexml.getPixelsSizeZ(0).getValue();
  int sizeT = omexml.getPixelsSizeT(0).getValue();
  int samplesPerChannel = omexml.getChannelSamplesPerPixel(0).getValue();
  sizeC /= samplesPerChannel;

  int imageCount = sizeC * sizeZ * sizeT;
```

(continues on next page)

(continued from previous page)

```
  for (int image=0; image<imageCount; image++) {
    writer.saveBytes(image, pixelData[image]);
  }
}
```

Finally, we must tell the writer that we are finished, so that the output file can be properly closed:

```
writer.close();
```

There should now be a complete OME-TIFF file at whichever path was specified above.

3.2.5 Tiled reading and writing in Bio-Formats

Reading tiled images

The reading of tiled images is straightforward and can be done in much the same way as reading a full image. In this case, to read an individual tile, we pass to the reader parameters for the x and y coordinates of the tile to read and the height and width of tile desired.

```
byte[] tile = reader.openBytes(image, tileX, tileY, tileSizeX, tileSizeY);
```

For TIFF-based readers, if the image has been written using tiles, then the tile width and height used can be found as below. These values can then be used with the above command to read the correct tiles individually.

```
IFD tileIFd = reader.getIFDs().get(0);
int tileHeight = tileIFd.getIFDIntValue(IFD.TILE_LENGTH);
int tileWidth = tileIFd.getIFDIntValue(IFD.TILE_WIDTH);
```

Introduction to tiled writing

Tiled writing is currently supported for TIFF-based formats. To set up an image writer to use tiling the following 2 API functions are provided:

```
public int setTileSizeX(int tileSize) throws FormatException
public int setTileSizeY(int tileSize) throws FormatException
```

Each function takes in an integer parameter for the desired tile size. As not all tile sizes are supported, the image writer will round the requested value to the nearest supported tile size. The return value will contain the actual tiling size which will be used by the writer.

To find out the tiling size currently being used at any point there are 2 further API functions to get the current tile size for a writer. If tiling is not being used or is not supported then the full image height and width will be returned.

```
public int getTileSizeX() throws FormatException
public int getTileSizeY() throws FormatException
```

The tiling parameters for writers must be set after the image metadata is set. An example of initializing a writer for tiling is shown below.

```
    // set up the writer and associate it with the output file
    writer = new OMETiffWriter();
    writer.setMetadataRetrieve(omexml);
    writer.setInterleaved(reader.isInterleaved());
```

(continues on next page)

(continued from previous page)

```
    // set the tile size height and width for writing
    this.tileSizeX = writer.setTileSizeX(tileSizeX);
    this.tileSizeY = writer.setTileSizeY(tileSizeY);

    writer.setId(outputFile);
```

Simple tiled writing

The simplest way to write a tiled image is to set the tiling parameters on your image writer as above and have the writer automatically handle the tiling. Once the tile sizes have been set you may simply read and write your image files as normal.

```
    // set up the writer and associate it with the output file
    writer = new OMETiffWriter();
    writer.setMetadataRetrieve(omexml);
    writer.setInterleaved(reader.isInterleaved());

    // set the tile size height and width for writing
    this.tileSizeX = writer.setTileSizeX(tileSizeX);
    this.tileSizeY = writer.setTileSizeY(tileSizeY);

    writer.setId(outputFile);
```

Full working example code is provided in

`SimpleTiledWriter.java` - code from that class is referenced here in part. You will need to have `bioformats_package.jar` in your Java `CLASSPATH` in order to compile `SimpleTiledWriter.java`.

Reading and writing using tiling

For the most efficient reading and writing of tiles you may instead wish to read in and write out the individual image tiles one at a time.

To do this you can set up the reader and writer as in the previous example above. In this case, when setting the tile height and width used it is important to store the return values which will be the valid tile size used by the writer.

```
// set the tile height and width and store the actual values used by the writer
int tileSizeX = writer.setTileSizeX(tileSizeX);
int tileSizeY = writer.setTileSizeY(tileSizeY);
```

This time for each image you must determined the number of tiles using the actual tile height and width being used.

```
        int width = reader.getSizeX();
        int height = reader.getSizeY();

        // Determined the number of tiles to read and write
        int nXTiles = width / tileSizeX;
        int nYTiles = height / tileSizeY;
        if (nXTiles * tileSizeX != width) nXTiles++;
        if (nYTiles * tileSizeY != height) nYTiles++;
```

Now each tile can be read and written individually.

```
        for (int y=0; y<nYTiles; y++) {
          for (int x=0; x<nXTiles; x++) {
            // The x and y coordinates for the current tile
            int tileX = x * tileSizeX;
            int tileY = y * tileSizeY;

            // Read tiles from the input file and write them to the output OME-Tiff
            buf = reader.openBytes(image, tileX, tileY, tileSizeX, tileSizeY);
            writer.saveBytes(image, buf, tileX, tileY, tileSizeX, tileSizeY);
          }
        }
```

Full working example code is provided in

`TiledReaderWriter.java` - code from that class is referenced here in part. You will need to have `bioformats_package.jar` in your Java `CLASSPATH` in order to compile `TiledReaderWriter.java`.

Tiles which overlap image size

It may not always be the case that the tile size divides evenly into the image height and width. In these scenarios in which the last column or row of tiles overlaps with the image boundary, a smaller tile is instead read or written for the final row or column. If a full sized tile with padding is written then a loci.formats.FormatException will be thrown for `Invalid tile size`.

To deal with this we can modify the previous tiled writing example to check if the tile size will overlap the image boundaries. If it will not overlap then the full tile size is used, if it does then the tile size for that tile is modified to reflect the partial tile.

```
            // If the last tile row or column overlaps the image size then only a␣
↪partial tile
            // is read or written. The tile size used is adjusted to account for any␣
↪overlap.
            int effTileSizeX = (tileX + tileSizeX) < width ? tileSizeX : width - tileX;
            int effTileSizeY = (tileY + tileSizeY) < height ? tileSizeY : height - tileY;

            // Read tiles from the input file and write them to the output OME-Tiff
            buf = reader.openBytes(image, tileX, tileY, effTileSizeX, effTileSizeY);
            writer.saveBytes(image, buf, tileX, tileY, effTileSizeX, effTileSizeY);
```

Full working example code is provided in

`OverlappedTiledWriter.java` - code from that class is referenced here in part. You will need to have `bioformats_package.jar` in your Java `CLASSPATH` in order to compile `OverlappedTiledWriter.java`.

3.2.6 Working with whole slide images

Bio-Formats supports many whole slide image formats, but effectively working with this type of data requires some extra considerations. Each file contains images representing one or more physical slides, with each slide typically stored at multiple resolutions. The width and height of a full resolution whole slide image often exceed 100,000 pixels, so the uncompressed image size may be several gigabytes. This means that only part of the full resolution image can be accessed at any given time.

JPEG or JPEG-2000 compression is typically used such that the size on disk is often less than 100MB. Most whole slide formats split each image into many small tiles of 1024×1024 pixels or smaller and compress each tile independently, though some (e.g. *Hamamatsu ndpi*) compress the whole image at once. Many supported whole slide formats are based upon TIFF, with vendor-specific extensions for metadata or tile storage. Notable exceptions include *Zeiss CZI* and *cellSens VSI*.

The original full resolution image and its resolutions are collectively referred to as an image pyramid. File formats which support image pyramids are noted by the `Pyramid` column in *the supported formats table*.

By default, openBytes will load from the full resolution image in the first pyramid stored in the file. Each resolution of each pyramid is stored as a separate series, and can be accessed by calling setSeries prior to retrieving pixel data.

There are additional API methods that can be used to make pyramids easier to work with. These can be enabled by calling setFlattenedResolutions(false) prior to setId.

After `setFlattenedResolutions(false)`, each series represents an entire image pyramid and not just a single resolution. Calling `setSeries(...)` then skips over all other resolutions in the same pyramid, to either the next pyramid (if multiple pyramids are stored), or the thumbnail or barcode image (if present). To access the smaller resolutions in the pyramid, use the getResolutionCount() and setResolution(int) methods.

Most formats only store one pyramid per fileset, but some (e.g. *cellSens VSI*) allow multiple pyramids. Almost all formats allow a thumbnail, slide overview, and/or slide barcode image. Bio-Formats always stores these images as separate series, after all of the pyramids. Be careful to check the pixel type for the extra images, as the type and channel count will often differ from that of the pyramid(s).

For an example of how to use the pyramid resolution API, see https://github.com/ome/bio-formats-examples/blob/master/src/main/java/SubResolutionExample.java.

Bio-Formats also provides some visibility into how the tiles are stored via the getOptimalTileWidth() and getOptimalTileHeight() methods. This is a suggestion of the size of tiles to be passed to openBytes(int, byte[], int, int, int, int), in order to minimize the number of tile decompressions. In most cases, and especially for the largest resolution, the whole image can't be loaded at once. The amount of memory allocated is not a factor in being able to load the whole image, as no more than 2GB of pixel data can be stored in a single byte array and most full resolution images will exceed this limit.

3.2.7 Pyramids in OME-TIFF

Bio-Formats 6.0.0 and later can read and write image pyramids in the OME-TIFF format. Reading OME-TIFF pyramids uses the same API as described above. Writing OME-TIFF pyramids requires the resolution dimensions to be specified in an `IPyramidStore` object. GeneratePyramidResolutions shows a simple example of how to do this.

The bfconvert command line tool will also automatically write pyramids if the input file has at least one pyramid, the output format is OME-TIFF, and the *`bfconvert -noflat`* option is specified.

3.2.8 Internal OMERO pyramid format

For files that contain very large images and are not in a format that supports pyramids, OMERO will generate its own image pyramid to improve visualization performance. Bio-Formats can read these generated pyramids, but cannot currently write them outside of OMERO. For details of how to read image pyramids with Bio-Formats, see *Working with whole slide images*

OMERO handles pyramid generation automatically for files that do not already have a stored pyramid, use a supported pixel type, and have images that exceed a specific XY size. The default XY size threshold is 3192×3192, but this can be configured in OMERO if necessary. Common formats for which a pyramid will be generated include *Gatan DM3*, *MRC*, and *TIFF*. Dedicated whole slide imaging formats such as *SVS* typically contain their own image pyramid, in which case an OMERO pyramid will not be generated.

For further information, see the OMERO pyramid specification.

3.2.9 In-memory reading and writing in Bio-Formats

Bio-Formats readers and writers are traditionally used to handle image files from disk. However it is also possible to achieve reading and writing of files from in-memory sources. This is handled by mapping the in-memory data to a file location using Location.mapFile().

```
Location.mapFile(fileName, byteArrayHandle);
```

This file location is not created on disk but rather maps internally to the in-memory data provided.

Reading file from memory

In order for Bio-Formats to read a file from memory it must be available in a byte array. For this example an input file is read from disk into a byte array.

```
    // read in entire file
    System.out.println("Reading file into memory from disk...");
    File inputFile = new File(path);
    int fileSize = (int) inputFile.length();
    DataInputStream in = new DataInputStream(new FileInputStream(inputFile));
    byte[] inBytes = new byte[fileSize];
    in.readFully(inBytes);
    System.out.println(fileSize + " bytes read.");
```

This data can now be handled by a Bio-Formats reader. This is achieved by providing a mapping from the in-memory data to a suitable filename which will be used by the reader. The filename used must have the same suffix as the original data type.

```
    // determine input file suffix
    String fileName = inputFile.getName();
    int dot = fileName.lastIndexOf(".");
    String suffix = dot < 0 ? "" : fileName.substring(dot);

    // map input id string to input byte array
    String inId = "inBytes" + suffix;
    Location.mapFile(inId, new ByteArrayHandle(inBytes));
```

Once the in-memory data has been mapped to a suitable filename the data can be handled by the reader as normal.

```
    // read in entire file
    System.out.println("Reading file into memory from disk...");
    File inputFile = new File(path);
    int fileSize = (int) inputFile.length();
    DataInputStream in = new DataInputStream(new FileInputStream(inputFile));
    byte[] inBytes = new byte[fileSize];
    in.readFully(inBytes);
    System.out.println(fileSize + " bytes read.");
```

Writing to memory

To use a writer to output to memory rather than an output file a similar process is required. First a mapping is created between a suitable output filename and the in-memory data.

```
    // map output id string to output byte array
    String outId = fileName + ".ome.tif";
    ByteArrayHandle outputFile = new ByteArrayHandle();
    Location.mapFile(outId, outputFile);
```

The mapped filename can now be passed to initialize the writer as standard.

```
// write data to byte array using ImageWriter
System.out.println();
System.out.print("Writing planes to destination in memory: ");
ImageWriter writer = new ImageWriter();
writer.setMetadataRetrieve(omeMeta);
writer.setId(outId);
```

The data can then be written to memory using the same read and write loop which would normally be used to write a file to disk.

```
byte[] plane = null;
for (int i=0; i<imageCount; i++) {
  if (plane == null) {
    // allow reader to allocate a new byte array
    plane = reader.openBytes(i);
  }
  else {
    // reuse previously allocated byte array
    reader.openBytes(i, plane);
  }
  writer.saveBytes(i, plane);
  System.out.print(".");
}
reader.close();
writer.close();
System.out.println();

byte[] outBytes = outputFile.getBytes();
outputFile.close();
```

If desired the data written to memory can then be flushed to disk and written to an output file location.

```
// flush output byte array to disk
System.out.println();
System.out.println("Flushing image data to disk...");
String outFileName = args.length > 1 ? args[1] : fileName + ".ome.tif";
File outFile = new File(outFileName);
DataOutputStream out = new DataOutputStream(new FileOutputStream(outFile));
out.write(outBytes);
out.close();
```

See also

`ReadWriteInMemory.java` - Full source code which is referenced here in part. You will need to have `bioformats_package.jar` in your Java `CLASSPATH` in order to compile `ReadWriteInMemory.java`.

3.2.10 Logging

Logging frameworks

Bio-Formats uses SLF4J as a logging API. SLF4J is a facade and needs to be bound to a logging framework at deployment time. Two underlying logging frameworks are currently supported by Bio-Formats. Bio-Formats does not

impose any specific SLF4J binding on downstream consumers as per the recommended SLF4J best practices. Some recommended bindings are as follows:

- logback is the recommended framework and natively implements the SLF4J API. A logback version of 1.2.x is recommended as Bio-Formats has a hard dependency for slf4j-api 1.7.30. This means that logback versions 1.3.x and higher are not yet supported in Bio-Formats as they rely on slf4j-api 2.x.
- log4j is the other logging framework supported by Bio-Formats and is primarily used in the *MATLAB environment*.

Examples of declaring these dependencies using Maven or Gradle are given in the Bio-Formats examples repository.

Initialization

The DebugTools class contains a series of framework-agnostic methods for the initialization and control of the logging system. This class uses reflection to detect the underlying logging framework and delegate the method calls to either Log4jTools or LogbackTools.

The main methods are described below:

- `DebugTools.enableLogging()` will initialize the underlying logging framework. This call will result in a no-op if logging has been initialized either via a binding-specific configuration file (see logback configuration) or via a prior call to `DebugTools.enableLogging()`.
- `DebugTools.enableLogging(level)` will initialize the logging framework under the same conditions as described above and set the root logger level if the initialization was succesful.
- `DebugTools.setRootLevel(level)` will override the level of the root logger independently of how the logging system was initialized.
- `DebugTools.enableIJLogging()` (logback-only) will add an ImageJ-specific appender to the root logger.

Changed in version 5.2.0: Prior to Bio-Formats 5.2.0, `DebugTools.enableLogging(level)` unconditionally set the logging and root logger level. Use `DebugTools.setRootLevel(level)` to restore this behavior.

3.2.11 Converting files from FV1000 OIB/OIF to OME-TIFF

This document explains how to convert a file from FV1000 OIB/OIF to OME-TIFF using Bio-Formats version 4.2 and later.

Working example code is provided in `FileConvert.java` - code from that class is referenced here in part. You will need to have `bioformats_package.jar` in your Java `CLASSPATH` in order to compile `FileConvert.java`.

The first thing that must happen is we must create the object that stores OME-XML metadata. This is done as follows:

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

The 'omexml' object can now be used by both a file format reader and a file format writer for storing and retrieving OME-XML metadata.

Now that have somewhere to put metadata, we need to create a file reader and writer:

```
ImageReader reader = new ImageReader();
ImageWriter writer = new ImageWriter();
```

Now we must associate the 'omexml' object with the file reader and writer:

```
reader.setMetadataStore(omexml);
writer.setMetadataRetrieve(omexml);
```

The reader now knows to store all of the metadata that it parses into 'omexml', and the writer knows to retrieve any metadata that it needs from 'omexml'.

We now tell the reader and writer which files will be read from and written to, respectively:

```
reader.setId("input-file.oib");
writer.setId("output-file.ome.tiff");
```

It is critical that the file name given to the writer ends with ".ome.tiff" or ".ome.tif", as it is the file name extension that determines which format will be written.

Now that everything is set up, we can convert the image data. This is done plane by plane:

```
for (int series=0; series<reader.getSeriesCount(); series++) {
  reader.setSeries(series);
  writer.setSeries(series);

  byte[] plane = new byte[FormatTools.getPlaneSize(reader)];
  for (int image=0; image<reader.getImageCount(); image++) {
    reader.openBytes(image, plane);
    writer.saveBytes(image, plane);
  }
}
```

The body of the outer 'for' loop may also be replaced with the following:

```
reader.setSeries(series);
writer.setSeries(series);

for (int image=0; image<reader.getImageCount(); image++) {
  byte[] plane = reader.openBytes(image);
  writer.saveBytes(image, plane);
}
```

But note that this will be a little slower.

Finally, we must tell the reader and writer that we are finished, so that the input and output files can be properly closed:

```
reader.close();
writer.close();
```

There should now be a complete OME-TIFF file at whichever path was specified above.

3.2.12 Using Bio-Formats in MATLAB

This section assumes that you have installed the MATLAB toolbox as instructed in the *MATLAB user information page*. Note the minimum recommended MATLAB version is R2017b.

As described in Using Java Libraries, every installation of MATLAB includes a JVM (Java Virtual Machine) allowing use of the Java API and third-party Java libraries. All the helper functions included in the MATLAB toolbox make use of the Bio-Formats Java API. Please refer to the Javadocs for more information.

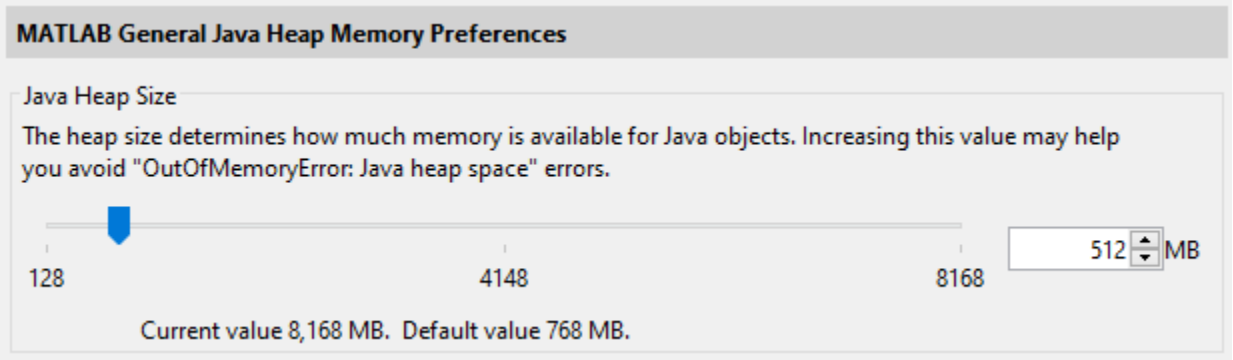
Increasing JVM memory settings

The default JVM settings in MATLAB can result in `java.lang.OutOfMemoryError: Java heap space` exceptions when opening large image files using Bio-Formats. Information about the Java heap space usage in MATLAB can be retrieved using:

```
java.lang.Runtime.getRuntime.maxMemory
```

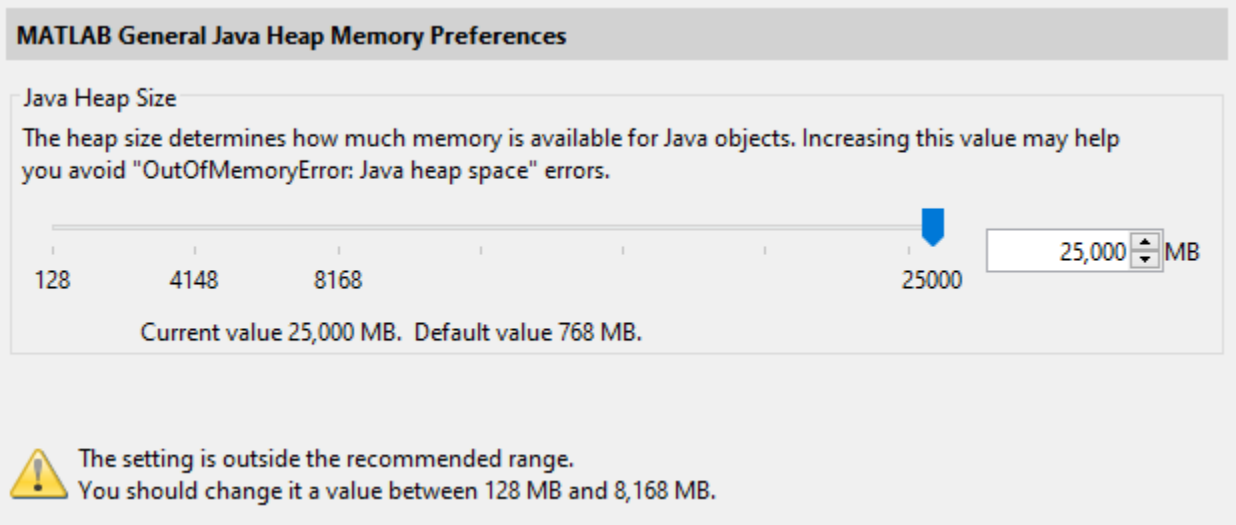
MATLAB Preferences (R2010a+)

Default JVM settings can be increased by Java Heap Memory Preferences of MATLAB (R2010a onwards) using *Preferences → General → Java Heap Memory*. We recommend to use 512 MB in general, but you don't need to decrease the value if the default value is bigger than 512 MB.



However, the maximum value allowed in the Preferences GUI (typically set to 25% of Physical Memory, whihc you can ask by the `memory` MATLAB command) can still be too small for your purpose. In that case, you can directly edit `matlab.prf` file in the folder specified by the `prefdir` MATLAB function (eg. `'C:\Users\xxxxxxx\AppData\Roaming\MathWorks\MATLAB\R2018b'`). Find the parameter `JavaMemHeapMax` in the file and increase the number that follows the capital `I` (in MB) to increase the maximum Java heap memory size. The change will be reflected by the Preferences as above.

```
JavaMemHeapMax=I25000
```



You will see a warning message as above.

java.opts

Alternatively, this can also be done by creating a `java.opts` file in the startup directory and overriding the default memory settings (see this page for more information). We recommend using `-Xmx512m` (meaning 512 MB) in your `java.opts` file. Calling:

```
bfCheckJavaMemory()
```

will also throw a warning if the runtime memory is lower than the recommended value.

If errors of type `java.lang.OutOfMemoryError: PermGen space` are thrown while using Bio-Formats with the Java bundled with MATLAB, you may try to increase the default values of `-XX:MaxPermSize` and `-XX:PermSize` via the `java.opts` file.

See also

Java Heap Memory Preferences (R2010a onwards)

How do I increase the heap space for the Java VM in MATLAB 6.0 (R12) and later versions?

Release of OMERO & Bio-Formats 5.1.1

Opening an image file

The first thing to do is initialize a file with the bfopen function:

```
data = bfopen('/path/to/data/file');
```

This function returns an `n`-by-4 cell array, where `n` is the number of series in the dataset. If `s` is the series index between 1 and `n`:

- The `data{s, 1}` element is an `m`-by-2 cell array, where `m` is the number of planes in the `s`-th series. If `t` is the plane index between 1 and `m`:
  - The `data{s, 1}{t, 1}` element contains the pixel data for the `t`-th plane in the `s`-th series.
  - The `data{s, 1}{t, 2}` element contains the label for the `t`-th plane in the `s`-th series.
- The `data{s, 2}` element contains original metadata key/value pairs that apply to the `s`-th series.
- The `data{s, 3}` element is an `m`-by-1 cell array, where `m` is the number of planes in the `s`-th series, which contains color lookup tables for each plane.
- The `data{s, 4}` element contains a standardized OME metadata structure, which is the same regardless of the input file format, and contains common metadata values such as physical pixel sizes - see *OME metadata* below for examples.

Changed in version 8.2.0.

Prior to Bio-Formats 8.2.0, the colormaps were returned as a `m`-by-`m` cell array for each series.

Accessing planes

Here is an example of how to unwrap specific image planes for easy access:

```
seriesCount = size(data, 1);
series1 = data{1, 1};
series2 = data{2, 1};
series3 = data{3, 1};
metadataList = data{1, 2};
% etc
series1_planeCount = size(series1, 1);
series1_plane1 = series1{1, 1};
series1_label1 = series1{1, 2};
```

(continues on next page)

(continued from previous page)

```
series1_plane2 = series1{2, 1};
series1_label2 = series1{2, 2};
series1_plane3 = series1{3, 1};
series1_label3 = series1{3, 2};
```

Displaying images

If you want to display one of the images, you can do so as follows:

```
series1_colorMap1 = data{1, 3}{1, 1};
figure('Name', series1_label1);
if isempty(series1_colorMap1)
  colormap(gray);
else
  colormap(series1_colorMap1);
end
imagesc(series1_plane1);
```

This will display the first image of the first series with its associated color map (if present). If you would prefer not to apply the color maps associated with each image, simply comment out the calls to `colormap`.

If you have the image processing toolbox, you could instead use:

```
imshow(series1_plane1, series1_colorMap1);
```

You can also create an animated movie (assumes 8-bit unsigned data):

```
cmap = gray(256);
for p = 1 : size(series1, 1)
  M(p) = im2frame(uint8(series1{p, 1}), cmap);
end
if feature('ShowFigureWindows')
  movie(M);
end
```

Retrieving metadata

There are two kinds of metadata:

- **Original metadata** is a set of key/value pairs specific to the input format of the data. It is stored in the `data{s, 2}` element of the data structure returned by `bfopen`.
- **OME metadata** is a standardized metadata structure, which is the same regardless of input file format. It is stored in the `data{s, 4}` element of the data structure returned by `bfopen`, and contains common metadata values such as physical pixel sizes, instrument settings, and much more. See the OME Model and Formats documentation for full details.

Original metadata

To retrieve the metadata value for specific keys:

```
% Query some metadata fields (keys are format-dependent)
metadata = data{1, 2};
```

(continues on next page)

(continued from previous page)

```
subject = metadata.get('Subject');
title = metadata.get('Title');
```

To print out all of the metadata key/value pairs for the first series:

```
metadataKeys = metadata.keySet().iterator();
for i=1:metadata.size()
  key = metadataKeys.nextElement();
  value = metadata.get(key);
  fprintf('%s = %s\n', key, value)
end
```

OME metadata

Conversion of metadata to the OME standard is one of Bio-Formats' primary features. The OME metadata is always stored the same way, regardless of input file format.

To access physical voxel and stack sizes of the data:

```
omeMeta = data{1, 4};
stackSizeX = omeMeta.getPixelsSizeX(0).getValue(); % image width, pixels
stackSizeY = omeMeta.getPixelsSizeY(0).getValue(); % image height, pixels
stackSizeZ = omeMeta.getPixelsSizeZ(0).getValue(); % number of Z slices

voxelSizeXdefaultValue = omeMeta.getPixelsPhysicalSizeX(0).value();           % returns␣
↪value in default unit
voxelSizeXdefaultUnit = omeMeta.getPixelsPhysicalSizeX(0).unit().getSymbol(); % returns␣
↪the default unit type
voxelSizeX = omeMeta.getPixelsPhysicalSizeX(0).value(ome.units.UNITS.MICROMETER); % in µm
voxelSizeXdouble = voxelSizeX.doubleValue();                                  % The␣
↪numeric value represented by this object after conversion to type double
voxelSizeY = omeMeta.getPixelsPhysicalSizeY(0).value(ome.units.UNITS.MICROMETER); % in µm
voxelSizeYdouble = voxelSizeY.doubleValue();                                  % The␣
↪numeric value represented by this object after conversion to type double
voxelSizeZ = omeMeta.getPixelsPhysicalSizeZ(0).value(ome.units.UNITS.MICROMETER); % in µm
voxelSizeZdouble = voxelSizeZ.doubleValue();                                  % The␣
↪numeric value represented by this object after conversion to type double
```

For more information about the methods to retrieve the metadata, see the MetadataRetrieve Javadoc page.

To convert the OME metadata into a string, use the `dumpXML()` method:

```
omeXML = char(omeMeta.dumpXML());
```

Changing the logging level

By default, `bfopen` uses `bfInitLogging` to initialize the logging system at the *WARN* level. To change the root logging level, use the DebugTools methods as described in the *Logging* section.

```
% Set the logging level to DEBUG
loci.common.DebugTools.setRootLevel('DEBUG');
```

Reading from an image file

The main inconvenience of the bfopen.m function is that it loads all the content of an image regardless of its size.

To access the file reader without loading all the data, use the low-level bfGetReader.m function:

```
reader = bfGetReader('path/to/data/file');
```

You can then access the OME metadata using the `getMetadataStore()` method:

```
omeMeta = reader.getMetadataStore();
```

Individual planes can be queried using the bfGetPlane.m function:

```
series1_plane1 = bfGetPlane(reader, 1);
```

To switch between series in a multi-image file, use the setSeries(int) method. To retrieve a plane given a set of *(z, c, t)* coordinates, these coordinates must be linearized first using getIndex(int, int, int)

```
% Read plane from series iSeries at Z, C, T coordinates (iZ, iC, iT)
% All indices are expected to be 1-based
reader.setSeries(iSeries - 1);
iPlane = reader.getIndex(iZ - 1, iC -1, iT - 1) + 1;
I = bfGetPlane(reader, iPlane);
```

Saving files

The basic code for saving a 5D array into an OME-TIFF file is located in the bfsave.m function.

For instance, the following code will save a single image of 64 pixels by 64 pixels with 8 unsigned bits per pixels:

```
plane = zeros(64, 64, 'uint8');
bfsave(plane, 'single-plane.ome.tiff');
```

And the following code snippet will produce an image of 64 pixels by 64 pixels with 2 channels and 2 timepoints:

```
plane = zeros(64, 64, 1, 2, 2, 'uint8');
bfsave(plane, 'multiple-planes.ome.tiff');
```

By default, `bfsave` will create a minimal OME-XML metadata object containing basic information such as the pixel dimensions, the dimension order and the pixel type. To customize the OME metadata, it is possible to create a metadata object from the input array using createMinimalOMEXMLMetadata.m, add custom metadata and pass this object directly to `bfsave`:

```
plane = zeros(64, 64, 1, 2, 2, 'uint8');
metadata = createMinimalOMEXMLMetadata(plane);
pixelSize = ome.units.quantity.Length(java.lang.Double(.05), ome.units.UNITS.MICROMETER);
metadata.setPixelsPhysicalSizeX(pixelSize, 0);
metadata.setPixelsPhysicalSizeY(pixelSize, 0);
pixelSizeZ = ome.units.quantity.Length(java.lang.Double(.2), ome.units.UNITS.MICROMETER);
metadata.setPixelsPhysicalSizeZ(pixelSizeZ, 0);
bfsave(plane, 'metadata.ome.tiff', 'metadata', metadata);
```

The dimension order of the file on disk can be specified in two ways:

- either by passing an OME-XML metadata option as a key/value pair as shown above
- or as an optional positional argument of `bfsave`

If a metadata object is passed to `bfsave`, its dimension order stored internally will take precedence.

For more information about the methods to store the metadata, see the MetadataStore Javadoc page.

Improving reading performance

Initializing a Bio-Formats reader can consume substantial time and memory. Most of the initialization time is spend in the setId(java.lang.String) call. Various factors can impact the performance of this step including the file size, the amount of metadata in the image and also the file format itself.

One solution to improve reading performance is to use Bio-Formats memoization functionalities with the loci.formats.Memoizer reader wrapper. By essence, the speedup gained from memoization will only happen after the first initialization of the reader for a particular file.

The simplest way to make use the `Memoizer` functionalities in MATLAB is illustrated by the following example:

```
% Construct an empty Bio-Formats reader
r = bfGetReader();
% Decorate the reader with the Memoizer wrapper
r = loci.formats.Memoizer(r);
% Initialize the reader with an input file
% If the call is longer than a minimal time, the initialized reader will
% be cached in a file under the same directory as the initial file
% name .large_file.bfmemo
r.setId(pathToFile);

% Perform work using the reader

% Close the reader
r.close()

% If the reader has been cached in the call above, re-initializing the
% reader will use the memo file and complete much faster especially for
% large data
r.setId(pathToFile);

% Perform additional work

% Close the reader
r.close()
```

If the time required to call setId(java.lang.String) method is larger than DEFAULT_MINIMUM_ELAPSED or the minimum value passed in the constructor, the initialized reader will be cached in a memo file under the same folder as the input file. Any subsequent call to `setId()` with a reader decorated by the `Memoizer` on the same input file will load the reader from the memo file instead of performing a full reader initialization.

More constructors are described in the Memoizer javadocs allowing to control the minimal initialization time required before caching the reader and/or to define a root directory under which the reader should be cached.

As Bio-Formats is not thread-safe, reader memoization offers a new solution to increase reading performance when doing parallel work. For instance, the following example shows how to combine memoization and MATLAB parfor to do work on a single file in a parallel loop:

```
% Construct a Bio-Formats reader decorated with the Memoizer wrapper
r = loci.formats.Memoizer(bfGetReader(), 0);
% Initialize the reader with an input file to cache the reader
```

(continues on next page)

(continued from previous page)

```
r.setId(pathToFile);
% Close reader
r.close()

nWorkers = 4;

% Enter parallel loop
parfor i = 1 : nWorkers
    % Initialize logging at INFO level
    bfInitLogging('INFO');
    % Initialize a new reader per worker as Bio-Formats is not thread safe
    r2 = javaObject('loci.formats.Memoizer', bfGetReader(), 0);
    % Initialization should use the memo file cached before entering the
    % parallel loop
    r2.setId(pathToFile);

    % Perform work

    % Close the reader
    r2.close()
end
```

3.2.13 Using Bio-Formats in Python

OME does not currently provide a Python implementation for Bio-Formats. However, there are several options you can use to read images from Python via Bio-Formats:

AICSImageIO

The AICSImageIO project includes support for Bio-Formats:

```
from aicsimageio import AICSImage
cells = AICSImage('/path/to/my/cells.ome.tif')

import napari
napari.view_image(cells.xarray_data)
napari.run()
```

PyImageJ

The PyImageJ project enables use of ImageJ2, which includes the SCIFIO library, which wraps Bio-Formats. In this way, you can open Bio-Formats-supported formats as NumPy arrays:

```
import imagej
ij = imagej.init('sc.fiji:fiji')
jcells = ij.io().open('/path/to/my/cells.ome.tif')
cells = ij.py.from_java(jcells)

import napari
napari.view_image(cells)
napari.run()
```

python-bioformats

The CellProfiler project has implemented a Python wrapper around Bio-Formats which can be installed using *pip*:

```
pip install python-bioformats
```

See also

https://pypi.org/project/python-bioformats
Source code of the CellProfiler Python wrapper for Bio-Formats

3.2.14 Using Bio-Formats in R

OME does not currently provide an R implementation for Bio-Formats. However, there are options provided by the community:

Bioconductor RBioFormats

The RBioFormats project from Bioconductor provides an R package which interfaces the OME Bio-Formats Java library to allow reading of proprietary microscopy image data and metadata.:

```
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

# The following initializes usage of Bioc devel
BiocManager::install(version='devel')

BiocManager::install("RBioFormats")


# To view documentation for the version of this package installed in your system
browseVignettes("RBioFormats")
```

For further details and examples of how to use the package, please see the API documentation

3.2.15 Using Bio-Formats in .NET

OME does not currently provide a .NET implementation for Bio-Formats. However, there are options you can use to read images from .NET via Bio-Formats:

Bioformats.NET6

The Bioformats.NET6 project provides Bio-Formats libraries converted to DLL, built with IKVM 8.7.1 Maven SDK for .NET6. To install add the below to your project file. Note that due to formatting the release version has extra whitespace surrounding it that can be removed :

```
<PropertyGroup>
  <MavenAdditionalRepositories>ome=https://artifacts.openmicroscopy.org/artifactory/
↪maven/;edu.ucar=https://maven.scijava.org/content/repositories/public/;</
↪MavenAdditionalRepositories>
</PropertyGroup>
<ItemGroup>
    <MavenReference Include="bioformats_package">
      <GroupId>ome</GroupId>
```

```
      <ArtifactId>bioformats_package</ArtifactId>
      <Version> 8.5.0 </Version>
    </MavenReference>
    <MavenReference Include="cdm-core">
      <GroupId>edu.ucar</GroupId>
      <ArtifactId>cdm-core</ArtifactId>
      <Version>5.3.3</Version>
    </MavenReference>
</ItemGroup>
```

3.2.16 Interfacing with Bio-Formats from non-Java code

Bio-Formats is written in Java, and is easiest to use with other Java code. However, it is possible to call Bio-Formats from a program written in another language. But how to do so depends on your program's needs.

Technologically, there are two broad categories of solutions: **in-process** approaches, and **inter-process** communication.

For details, see LOCI's article, Interfacing from non-Java code.

Example **in-process solution**: Bio-Formats JACE C++ bindings (note that this is a legacy project and no longer actively maintained).

See also

Additional reader and writer options

3.3 Using Bio-Formats as a native C++ library

Note

See the OME-Files C++ downloads page for more information.

3.4 Contributing to Bio-Formats

3.4.1 Code formatting

Note, these guidelines do not cover:

- third-party code imported into the source tree, which is covered by the guidelines for the upstream projects
- released schema files which would require re-releasing if changed by reindenting

All languages

- Use spaces to indent; do not ever use tabs

Java

All Java code is formatted with:

- an indentation size of two spaces
- braces use the Java variant of K&R style

XML

All XML code is formatted with:

- an indentation size of two spaces
- attributes on multiple lines aligned vertically after the element name.

3.4.2 Testing code changes

Automated tests

The Bio-Formats testing framework component contains the infrastructure to run automated tests against reference data. These tests check the metadata and binary data of every dataset using INI-based configuration files as the source of truth.

All representative files submitted as part of the *bug report* or development projects and used as supporting data for new Bio-Formats releases are stored in a curated QA repository, also known as the full data repository, managed by the Bio-Formats maintainers alongside the associated configuration files. As part of the review process, every change against Bio-Formats is tested nightly against the entire data repository and all tests must be passing in order for a contribution to get integrated.

It is also possible for contributors to generate their configuration files and run the automated tests against their own data. Assuming the data is stored under `/path/to/data`, after checking out source code and building all the JAR files (see *Obtaining and building Bio-Formats*), switch to the `test-suite` component and run the **`ant`** `gen-config` target to generate the configuration files under `/path/to/config`:

```
$ cd components/test-suite
$ ant -Dtestng.directory=/path/to/data -Dtestng.configDirectory=/path/to/config gen-
↪config
```

To run the automated tests using existing configuration files, use the **`ant`** `test-automated` target:

```
$ cd components/test-suite
$ ant -Dtestng.directory=/path/to/data -Dtestng.configDirectory=/path/to/config test-
↪automated
```

Multiple options can be passed to the **`ant`** `test-automated` target by setting the `testng.${option}` option via the command line. Useful options are described below.

testng.directory
: Mandatory option. Specifies the root of the data directory to be tested:

```
$ ant -Dtestng.directory=/path/to/data/dataset1 -Dtestng.configDirectory=/path/to/
↪config/dataset1 test-automated
```

On Windows, the arguments to the test command must be quoted:

```
> ant "-Dtestng.directory=C:\\path\to\data\dataset1" "-Dtestng.configDirectory=C:\\
↪path\to\config\dataset1" test-automated
```

testng.configDirectory
: Mandatory option. Specifies the root of the directory containing the configuration files. This directory must have the same hierarchy as the one specified by `testng.directory` and contain `.bioformats` configuration files:

```
$ ant -Dtestng.directory=/path/to/data -Dtestng.configDirectory=/path/to/config␣
↪test-automated
```

testng.configSuffix

Specifies an optional suffix for the configuration files:

```
$ ant -Dtestng.directory=/path/to/data -Dtestng.configDirectory=/path/to/config -
↪Dtestng.configSuffix=win test-automated
```

testng.memory

Specifies the amount of memory to be allocated to the JVM:

```
$ ant -Dtestng.directory=/path/to/data -Dtestng.configDirectory=/path/to/config -
↪Dtestng.memory=4g test-automated
```

Default: 512m.

testng.threadCount

Specifies the number of threads to use for testing:

```
$ ant -Dtestng.directory=/path/to/data -Dtestng.configDirectory=/path/to/config -
↪Dtestng.threadCount=4 test-automated
```

Default: 1.

You should now see output similar to this:

```
Buildfile: build.xml

init-title:
     [echo] ----------=========== bio-formats-testing-framework ===========----------
...
test-automated:
   [testng] 17:05:28,713 |-INFO in ch.qos.logback.classic.LoggerContext[default] - Could␣
↪NOT find resource [${logback.configurationFile}]
   [testng] 17:05:28,713 |-INFO in ch.qos.logback.classic.LoggerContext[default] - Could␣
↪NOT find resource [logback.groovy]
   [testng] 17:05:28,713 |-INFO in ch.qos.logback.classic.LoggerContext[default] - Could␣
↪NOT find resource [logback-test.xml]
   [testng] 17:05:28,713 |-INFO in ch.qos.logback.classic.LoggerContext[default] - Found␣
↪resource [logback.xml] at [file:/opt/ome/bioformats/components/test-suite/logback.xml]
   [testng] 17:05:28,835 |-INFO in ch.qos.logback.core.joran.action.AppenderAction -␣
↪About to instantiate appender of type [ch.qos.logback.core.ConsoleAppender]
   [testng] 17:05:28,837 |-INFO in ch.qos.logback.core.joran.action.AppenderAction -␣
↪Naming appender as [stdout]
   [testng] 17:05:28,876 |-INFO in ch.qos.logback.core.joran.action.AppenderAction -␣
↪About to instantiate appender of type [ch.qos.logback.classic.sift.SiftingAppender]
   [testng] 17:05:28,878 |-INFO in ch.qos.logback.core.joran.action.AppenderAction -␣
↪Naming appender as [SIFT]
   [testng] 17:05:28,891 |-INFO in ch.qos.logback.classic.joran.action.LoggerAction -␣
↪Setting level of logger [loci.tests.testng] to DEBUG
   [testng] 17:05:28,891 |-INFO in ch.qos.logback.classic.joran.action.RootLoggerAction -
↪ Setting level of ROOT logger to INFO
   [testng] 17:05:28,891 |-INFO in ch.qos.logback.core.joran.action.AppenderRefAction -␣
↪Attaching appender named [SIFT] to Logger[ROOT]
   [testng] 17:05:28,892 |-INFO in ch.qos.logback.core.joran.action.AppenderRefAction -␣
↪Attaching appender named [stdout] to Logger[loci.tests.testng]
   [testng] 17:05:28,892 |-INFO in ch.qos.logback.classic.joran.action.
```

(continues on next page)

(continued from previous page)

```
↪ConfigurationAction - End of configuration.
   [testng] 17:05:28,894 |-INFO in ch.qos.logback.classic.joran.
↪JoranConfigurator@706a04ae - Registering current configuration as safe fallback point
   [testng] [2015-08-18 17:05:28,904] [main] testng.directory = /ome/data_repo/test_per_
↪commit/
   [testng] 17:05:28,908 |-INFO in ch.qos.logback.core.joran.action.AppenderAction -␣
↪About to instantiate appender of type [loci.tests.testng.TimestampedLogFileAppender]
   [testng] 17:05:28,909 |-INFO in ch.qos.logback.core.joran.action.AppenderAction -␣
↪Naming appender as [logfile-main]
   [testng] 17:05:28,955 |-INFO in loci.tests.testng.TimestampedLogFileAppender[logfile-
↪main] - File property is set to [target/bio-formats-test-main-2015-08-18_17-05-28.log]
   [testng] [2015-08-18 17:05:28,963] [main] testng.multiplier = 1.0
   [testng] [2015-08-18 17:05:28,964] [main] testng.in-memory = false
   [testng] [2015-08-18 17:05:28,964] [main] user.language = en
   [testng] [2015-08-18 17:05:28,964] [main] user.country = US
   [testng] [2015-08-18 17:05:28,964] [main] Maximum heap size = 455 MB
   [testng] Scanning for files...
   [testng] [2015-08-18 17:05:32,258] [main] ----------------------------------------
   [testng] [2015-08-18 17:05:32,258] [main] Total files: 480
   [testng] [2015-08-18 17:05:32,258] [main] Scan time: 3.293 s (6 ms/file)
   [testng] [2015-08-18 17:05:32,258] [main] ----------------------------------------
   [testng] Building list of tests...
```

and then eventually:

```
   [testng] ================================================
   [testng] Bio-Formats software test suite
   [testng] Total tests run: 19110, Failures: 0, Skips: 0
   [testng] ================================================
   [testng]

BUILD SUCCESSFUL
Total time: 16 minutes 42 seconds
```

In most cases, test failures should be logged in the main console output as:

```
[testng] [2015-08-18 17:13:13,625] [pool-1-thread-1]     SizeZ: FAILED (Series 0␣
↪(expected 2, actual 1))
```

To identify the file, look for the initialization line preceding the test failures under the same thread:

```
[testng] [2015-08-18 17:13:12,376] [pool-1-thread-1] Initializing /ome/data_repo/test_
↪per_commit/ome-tiff/img_bk_20110701.ome.tif:
```

The console output is also recorded under `components/test-suite/target` as `bio-formats-software-test-main-$DATE.log` where "$DATE" is the date on which the tests started in "yyyy-MM-dd_hh-mm-ss" format. The detailed report of each thread is recorded under `bio-formats-software-pool-$POOL-thread-$THREAD-main-$DATE.log`

MATLAB tests

Tests for the Bio-Formats MATLAB toolbox are written using the xunit framework and are located under https://github.com/ome/bioformats/tree/develop/components/formats-gpl/test/matlab.

To run these tests, you will need to download or clone matlab-xunit, a xUnit framework with JUnit-compatible XML output. Then add this package together with the Bio-Formats MATLAB to your MATLAB path:

```
% Add the matlab-xunit toolbox to the MATLAB path
addpath('/path/to/matlab-xunit');
% Add the Bio-Formats MATLAB source to the MATLAB path
% For developers working against the source code
addpath('/path/to/bioformats/components/formats-gpl/matlab');
addpath('/path/to/bioformats/artifacts');
% For developers working against a built artifact, e.g. a release
% addpath('/path/to/bfmatlab');
```

You can run all the MATLAB tests using **runxunit**:

```
cd /path/to/bioformats/components/formats-gpl/test/matlab
runxunit
```

Individual test classes can be run by passing the name of the class:

```
cd /path/to/bioformats/components/formats-gpl/test/matlab
runxunit TestBfsave
```

Individual test methods can be run by passing the name of the class and the name of the method:

```
cd /path/to/bioformats/components/formats-gpl/test/matlab
runxunit TestBfsave:testLZW
```

Finally, to output the test results under XML format, you can use the `-xmlfile` option:

```
cd /path/to/bioformats/components/formats-gpl/test/matlab
runxunit -xmlfile test-output.xml
```

3.4.3 Generating test images

Sometimes it is nice to have a file of a specific size or pixel type for testing. Bio-Formats defines an internal format used for generating test images. Test images recognized by Bio-Formats:

- must have an extension of type *.fake*
- must encode the image metadata using key-value pairs separated by = either in the filename or in a companion file
- may be accompanied by an INI-style companion file. A companion file must use the same basename as the fake file and be suffixed with *.ini*

Filename format

To generate an image file (that contains a gradient image):

```
touch "my-special-test-file&pixelType=uint8&sizeX=8192&sizeY=8192.fake"
```

Whatever is before the first & is the image name; the remaining key-value pairs, each preceded with &, set the pixel type and image dimensions. Just replace the values with whatever you need for testing. See *Key-value pairs* for the full description of available key/value pairs.

Companion file

You can put metadata values in a separate UTF-8 encoded `.fake.ini` file with the same basename as the fake file:

```
touch my-special-test-file.fake
echo "pixelType=uint8" >> my-special-test-file.fake.ini
echo "sizeX=8192" >> my-special-test-file.fake.ini
echo "sizeY=8192" >> my-special-test-file.fake.ini
```

In fact, just the `.fake.ini` file alone suffices:

```
echo "pixelType=uint8" >> my-special-test-file.fake.ini
echo "sizeX=8192" >> my-special-test-file.fake.ini
echo "sizeY=8192" >> my-special-test-file.fake.ini
```

If you include a "[GlobalMetadata]" section to the ini file, then all the included values will be accessible from the global metadata map:

```
echo "[GlobalMetadata]" >> my-special-test-file.fake.ini
echo "my.key=some.value" >> my-special-test-file.fake.ini
```

The `.ini` file can also contain one section for each series, which allows metadata such as exposure times and positions to be set for each plane:

```
echo "[series_0]" >> my-special-test-file.fake.ini
echo "ChannelEmissionWavelength_0=461nm" >> my-special-test-file.fake.ini
echo "ChannelExcitationWavelength_0=358nm" >> my-special-test-file.fake.ini
echo "ExposureTimeUnit_0=ms" >> my-special-test-file.fake.ini
echo "ExposureTime_0=10" >> my-special-test-file.fake.ini
echo "ExposureTimeUnit_0=ms" >> my-special-test-file.fake.ini
echo "PositionX_0=5" >> my-special-test-file.fake.ini
echo "PositionY_0=-5" >> my-special-test-file.fake.ini
echo "PositionZ_0=1" >> my-special-test-file.fake.ini
echo "PositionXUnit_0=mm" >> my-special-test-file.fake.ini
echo "PositionYUnit_0=mm" >> my-special-test-file.fake.ini
echo "PositionZUnit_0=mm" >> my-special-test-file.fake.ini
```

Several keys have support for units and can be expressed as `KEY=VALUE UNIT` where `UNIT` is the symbol of the desired unit:

```
touch "physicalSizesUnits&physicalSizeX=1nm&physicalSizeY=1nm&physicalSizeZ=1.5km.fake"
echo "physicalSizeX=1 nm" >> physicalSizes.fake.ini
echo "physicalSizeY=10 pm" >> physicalSizes.fake.ini
echo "physicalSizeZ=.002 mm" >> physicalSizes.fake.ini
```

High-content screening

To generate a simple plate file:

```
touch "simple-plate&plates=1&plateAcqs=1&plateRows=1&plateCols=1&fields=1.fake"
touch "default-plate&plates=1.fake"
touch "default-plate&screens=0&plates=1.fake"
```

These will each create a single plate without a containing screen, by default in the first two cases. In the third case setting `screens` to zero is used to document the lack of a screen. As above a `.fake.ini` file can be used.

To generate a simple screen file:

```
touch "default-screen&screens=1.fake"
```

This will create a screen containing a single simple plate.

To generate a valid plate at least one of `screens`, `plates`, `plateAcqs`, `plateRows`, `plateCols` and `fields` must be greater than zero. If this condition is met then any other plate-specific values set to zero will be ignored and the defaults used. So, for example, the file:

```
one-key-set&screens=0&plates=0&plateRows=0&plateCols=0&plateAcqs=0&fields=1.fake
```

will create a simple plate with no screen.

Regions

To generate a fake file containing regions of interest:

```
touch "regions&points=10.fake"
touch "regions&ellipses=20.fake"
touch "regions&rectangles=5&lines=25.fake"
```

Replace `regions` in the above examples with the desired image or plate which will contain the regions, e.g.

```
touch "HCSanalysis&plates=1&plateRows=16&plateCols=24&rectangles=100.fake"
```

For each shape type, the value will specify the number of regions of interest to create where each region of interest contains a single shape of the input type. By convention, all generated regions of interests are not associated to any given Z, C or T plane.

Sub-resolutions

Added in version 6.0.0.

To generate a fake file containing sub-resolutions:

```
touch "pyramid1&sizeX=20000&sizeY=10000&resolutions=8.fake"
touch "pyramid2&sizeX=20000&sizeY=10000&resolutions=4&resolutionScale=4.fake"
```

The `resolutions` and `resolutionScale` specify the number of sub-resolutions for each plane and the downsampling factor between consecutive sub-resolutions.

Key-value pairs

There are several other keys that can be added, a complete list of these, with their default values, is shown below.

Key	Value	Default
sizeX	number of pixels wide	512
sizeY	number of pixels tall	512
sizeZ	number of Z sections	1
sizeC	number of channels	1
sizeT	number of timepoints	1
thumbSizeX	number of pixels wide, for the thumbnail	0
thumbSizeY	number of pixels tall, for the thumbnail	0
pixelType	pixel type	uint8
bitsPerPixel	number of valid bits (<= number of bits implied by pixel type)	0
rgb	number of channels that are merged together	1
dimOrder	dimension order (e.g. XYZCT)	XYZCT
orderCertain	whether or not the dimension order is certain	true
little	whether or not the pixel data should be little-endian	true
interleaved	whether or not merged channels are interleaved	false
indexed	whether or not a color lookup table is present	false
falseColor	whether or not the color lookup table is just for making the image look pretty	false
metadataComplete	whether or not the metadata is complete	true
thumbnail	whether or not `CoreMetadata.thumbnail` is set	false
series	number of series (Images)	1
lutLength	number of entries in the color lookup table	3
scaleFactor	the scaling factor for the pixel values on each plane	1
exposureTime	time of exposure	null
acquisitionDate	timestamp formatted as "yyyy-MM-dd_HH-mm-ss"	null
screens	number of screens	0
plates	number of plates to generate	0[1]
plateAcqs	number of plate runs	0[Page 81, 1]
plateRows	number of rows per plate	0[Page 81, 1]
plateCols	number of columns per plate	0[Page 81, 1]
fields	number of fields per well	0[Page 81, 1]
withInstrument	whether or not a populated instrument should be added to the metadata	false
withMicrobeam	whether or not a microbeam should be added to the experiment (HCS only)	false
annLong, annDouble, annMap, annComment, annBool, annTime, annTag, annTerm, annXml	number of annotations of the given type to generate	0
physicalSizeX	real width of the pixels, supports units defaulting to microns	
physicalSizeY	real height of the pixels, supports units defaulting to microns	
physicalSizeZ	real depth of the pixels, supports units defaulting to microns	
color	the default color for all channels[3]	null
color_x	the color for channel x, overrides the default color for that channel[Page 81, 3]	
ellipses, labels, lines, points, polygons, polylines, rectangles	the number of ROIs containing one shape of the given type to generate	
emission_x	the emission wavelength for channel `x`, supports units defaulting to nanometers	
excitation_x	the excitation wavelength for channel `x`, supports units defaulting to nanometers	

continues on next page

Table 2 – continued from previous page

Key	Value	Default
ChannelEmissionWavelength_x	the emission wavelength for channel `x`, supports units defaulting to nanometers[2]	
ChannelExcitationWavelength_x	the excitation wavelength for channel `x`, supports units defaulting to nanometers[2]	
ChannelName_x	the channel name for channel `x`[2]	
DeltaT_x	time since the beginning of the acquisition for plane `x`[2]	
DeltaTUnit_x	string defining the units for the corresponding `DeltaT_x`[2]	seconds
ExposureTime_x	floating point exposure time for plane `x`[2]	
ExposureTimeUnit_x	string defining the units for the corresponding `ExposureTime_x`[2]	seconds
PositionX_x	floating point X position for plane `x`[2]	
PositionXUnit_x	string defining the units for the corresponding `PositionX_x`[2]	microns
PositionY_x	floating point Y position for plane `x`[2]	
PositionYUnit_x	string defining the units for the corresponding `PositionY_x`[2]	microns
PositionZ_x	floating point Z position for plane `x`[2]	
PositionZUnit_x	string defining the units for the corresponding `PositionZ_x`[2]	microns
resolutions	number of pyramid levels or sub-resolutions for each series	1
resolutionScale	for images with sub-resolutions, scaling factor between consecutive pyramid levels	2
sleepOpenBytes	number of milliseconds to sleep for when openBytes is called	0
sleepInitFile	number of milliseconds to sleep for when initFile is called	0
labelPlanes	add human-readable labels to the individual planes	false

For full details of these keys, how unset and default values are handled and further examples see loci.formats.in.FakeReader.

You can often work with the .fake file directly, but in some cases support for those files is disabled and so you will need to convert the file to something else. Make sure that you have Bio-Formats built and the JARs in your `CLASSPATH` (individual JARs or just `bioformats_package.jar`):

```
bfconvert test&pixelType=uint8&sizeX=8192&sizeY=8192.fake test.tiff
```

If you do not have the command line tools installed, substitute loci.formats.tools.ImageConverter for **`bfconvert`**.

3.4.4 Writing a new file format reader

This document is a brief guide to writing new Bio-Formats file format readers.

All format readers should extend either loci.formats.FormatReader or an existing reader.

Methods to override

- isSingleFile(java.lang.String) Whether or not the named file is expected to be the only file in the dataset. This only needs to be overridden for formats whose datasets can contain more than one file.
- isThisType(loci.common.RandomAccessInputStream) Check the first few bytes of a file to determine if the file can be read by this reader. You can assume that index 0 in the stream corresponds to the index 0 in the file. Return true if the file can be read; false if not (or if there is no way of checking).

[1] Default value set to 1 if any of the `screens`, `plates`, `plateAcqs`, `plateRows`, `plateCols` or `fields` values is set to a value greater than zero.

[3] Colors are specified as a single packed integer representing an RGBA value. This can be a decimal value (e.g. `16711935` for green with alpha = 255) or a hexadecimal value (e.g. `0x00FF00FF` for green with alpha = 255).

[2] Must be stored in the INI file under a `[series_n]` section, where `n` is the 0-based series index.

- fileGroupOption(java.lang.String) Returns an indication of whether or not the files in a multi-file dataset can be handled individually. The return value should be one of the following:
  - FormatTools.MUST_GROUP: the files cannot be handled separately
  - FormatTools.CAN_GROUP: the files may be handled separately or as a single unit
  - FormatTools.CANNOT_GROUP: the files must be handled separately

  This method only needs to be overridden for formats whose datasets can contain more than one file.
- getSeriesUsedFiles(boolean) You only need to override this if your format uses multiple files in a single dataset. This method should return a list of all files associated with the given file name and the current series (i.e. every file needed to display the current series). If the `noPixels` flag is set, then none of the files returned should contain pixel data. For an example of how this works, see loci.formats.in.PerkinElmerReader. It is recommended that the first line of this method be `FormatTools.assertId(currentId, true, 1)` - this ensures that the file name is non-null.
- openBytes(int, byte[], int, int, int, int) Returns a byte array containing the pixel data for a specified subimage from the given file. The dimensions of the subimage (upper left X coordinate, upper left Y coordinate, width, and height) are specified in the final four int parameters. This should throw a FormatException if the image number is invalid (less than 0 or >= the number of images). The ordering of the array returned by openBytes should correspond to the values returned by isLittleEndian and isInterleaved. Also, the length of the byte array should be [image width * image height * bytes per pixel]. Extra bytes will generally be truncated. It is recommended that the first line of this method be `FormatTools.checkPlaneParameters(this, no, buf.length, x, y, w, h)` - this ensures that all of the parameters are valid.
- initFile(java.lang.String) The majority of the file parsing logic should be placed in this method. The idea is to call this method once (and only once!) when the file is first opened. Generally, you will want to start by calling `super.initFile(String)`. You will also need to set up the stream for reading the file, as well as initializing any dimension information and metadata. Most of this logic is up to you; however, you should populate the core variable (see loci.formats.CoreMetadata).

  Note that each variable is initialized to 0 or null when `super.initFile(String)` is called. Also, `super.initFile(String)` constructs a Hashtable called metadata where you should store any relevant metadata.

  The most common way to set up the OME-XML metadata for the reader is to initialize the MetadataStore using the makeFilterMetadata() method and populate the Pixels elements of the metadata store from the `core` variable using the MetadataTools.populatePixels(MetadataStore, FormatReader) method:

```
# Initialize the OME-XML metadata from the core variable
MetadataStore store = makeFilterMetadata();
MetadataTools.populatePixels(store, this);
```

  If the reader includes metadata at the plane level, you can initialize the Plane elements under the Pixels using MetadataTools.populatePixels(MetadataStore, FormatReader, doPlane):

```
MetadataTools.populatePixels(store, this, true);
```

  Once the metadatastore has been initialized with the core properties, additional metadata can be added to it using the setter methods. Note that for each of the model components, the `setObjectID()` method should be called before any of the `setObjectProperty()` methods, e.g.:

```
# Add an oil immersion objective with achromat
String objectiveID = MetadataTools.createLSID("Objective", 0, 0);
store.setObjectiveID(objectiveID, 0, 0);
store.setObjectiveImmersion(getImmersion("Oil"), 0, 0);
```

- close(boolean) Cleans up any resources used by the reader. Global variables should be reset to their initial state, and any open files or delegate readers should be closed.

Note that if the new format is a variant of a format currently supported by Bio-Formats, it is more efficient to make the new reader a subclass of the existing reader (rather than subclassing loci.formats.FormatReader). In this case, it is usually sufficient to override initFile(java.lang.String) and isThisType(byte[]).

Every reader also has an instance of loci.formats.CoreMetadata. All readers should populate the fields in CoreMetadata, which are essential to reading image planes.

If you read from a file using something other than loci.common.RandomAccessInputStream or loci.common.Location, you *must* use the file name returned by `Location.getMappedId(String)`, not the file name passed to the reader. Thus, a stub for `initFile(String)` might look like this:

```
protected void initFile(String id) throws FormatException, IOException {
  super.initFile(id);

  RandomAccessInputStream in = new RandomAccessInputStream(id);
  // alternatively,
  //FileInputStream in = new FileInputStream(Location.getMappedId(id));

  // read basic file structure and metadata from stream
}
```

For more details, see loci.common.Location.mapId(java.lang.String, java.lang.String) and loci.common.Location.getMappedId(java.lang.String).

Variables to populate

There are a number of global variables defined in loci.formats.FormatReader that should be populated in the constructor of any implemented reader.

These variables are:

- suffixNecessary Indicates whether or not a file name suffix is required; true by default
- suffixSufficient Indicates whether or not a specific file name suffix guarantees that this reader can open a particular file; true by default
- hasCompanionFiles Indicates whether or not there is at least one file in a dataset of this format that contains only metadata (no images); false by default
- datasetDescription A brief description of the layout of files in datasets of this format; only necessary for multi-file datasets
- domains An array of imaging domains for which this format is used. Domains are defined in loci.formats.FormatTools.

Other useful things

- loci.common.RandomAccessInputStream is a hybrid RandomAccessFile/InputStream class that is generally more efficient than either RandomAccessFile or InputStream, and implements the DataInput interface. It is recommended that you use this for reading files.
- loci.common.Location provides an API similar to java.io.File, and supports File-like operations on URLs. It is highly recommended that you use this instead of File. See the Javadocs for additional information.
- loci.common.DataTools provides a number of methods for converting bytes to shorts, ints, longs, etc. It also supports reading most primitive types directly from a RandomAccessInputStream (or other DataInput implementation).

- loci.formats.ImageTools provides several methods for manipulating primitive type arrays that represent images. Consult the source or Javadocs for more information.
- If your reader relies on third-party code which may not be available to all users, it is strongly suggested that you make a corresponding service class that interfaces with the third-party code. Please see *Bio-Formats service and dependency infrastructure* for a description of the service infrastructure, as well as the loci.formats.services package.
- Several common image compression types are supported through subclasses of loci.formats.codec.BaseCodec. These include JPEG, LZW, LZO, Base64, ZIP and RLE (PackBits).
- If you wish to convert a file's metadata to OME-XML (strongly encouraged), please see *Bio-Formats metadata processing* for further information.
- Once you have written your file format reader, add a line to the readers.txt file with the fully qualified name of the reader, followed by a '#' and the file extensions associated with the file format. Note that loci.formats.ImageReader, the master file format reader, tries to identify which format reader to use according to the order given in readers.txt, so be sure to place your reader in an appropriate position within the list.

See also

External file format readers

- The easiest way to test your new reader is by calling "java loci.formats.tools.ImageInfo <file name>". If all goes well, you should see all of the metadata and dimension information, along with a window showing the images in the file. loci.formats.ImageReader can take additional parameters; a brief listing is provided below for reference, but it is recommended that you take a look at the contents of loci.formats.tools.ImageInfo to see exactly what each one does.

Argument	Action
-version	print the library version and exit
file	the image file to read
-nopix	read metadata only, not pixels
-nocore	do not output core metadata
-nometa	do not parse format-specific metadata table
-nofilter	do not filter metadata fields
-thumbs	read thumbnails instead of normal pixels
-minmax	compute min/max statistics
-merge	combine separate channels into RGB image
-nogroup	force multi-file datasets to be read as individual files
-stitch	stitch files with similar names
-separate	split RGB image into separate channels
-expand	expand indexed color to RGB
-omexml	populate OME-XML metadata
-normalize	normalize floating point images*
-fast	paint RGB images as quickly as possible*
-debug	turn on debugging output
-range	specify range of planes to read (inclusive)
-series	specify which image series to read
-swap	override the default input dimension order
-shuffle	override the default output dimension order
-map	specify file on disk to which name should be mapped
-preload	pre-read entire file into a buffer; significantly reduces the time required to read the images, but requires more memory
-crop	crop images before displaying; argument is 'x,y,w,h'
-autoscale	used in combination with '-fast' to automatically adjust brightness and contrast
-novalid	do not perform validation of OME-XML
-omexml-only	only output the generated OME-XML
-format	read file with a particular reader (e.g., ZeissZVI)

* = may result in loss of precision

- If you wish to test using TestNG, `loci.tests.testng.FormatReaderTest` provides several basic tests that work with all Bio-Formats readers. See the FormatReaderTest source code for additional information.
- For more details, please look at the source code and Javadocs. Studying existing readers is probably the best way to get a feel for the API; we would recommend first looking at loci.formats.in.ImarisReader (this is the most straightforward one). loci.formats.in.LIFReader and InCellReader are also good references that show off some of the nicer features of Bio-Formats.

If you have questions about Bio-Formats, please contact the forums.

3.4.5 External file format readers

File format readers recognized by Bio-Formats can be defined as either "core" or "external". A core reader is maintained by the OME team, and included in either the *formats-bsd* or *formats-gpl* component in the *ome/bioformats* repository on GitHub. An external reader is not included in the *ome/bioformats* repository, and in most cases is not maintained by the OME team.[1]

[1] A notable exception is ZarrReader, which is an external reader maintained by the OME team. This reader is externalized primarily to allow for updates outside of the normal Bio-Formats release process. See further discussion in this issue.

Bio-Formats contains a file named readers.txt which records every recognized reader class name. This file is used by the main reader class (loci.formats.ImageReader) to find installed readers.

For readers that are included in core Bio-Formats (i.e. the *ome/bioformats* repository on GitHub), a line containing just the fully qualified class name is added, e.g.:

```
loci.formats.in.OMETiffReader        # tif
```

An optional comment can be added after the class name using #.

Readers that are not in *ome/bioformats* and which are not included in Bio-Formats releases can be defined as "external" readers, e.g.:

```
loci.formats.in.ZarrReader[type=external]        # .zarr
```

This allows Bio-Formats to attempt to load a reader that was installed separately, but quietly ignore if it is not found.

When to use an external reader

The OME team often encourages new readers to be proposed as "external" under any of the following conditions:

- the newly introduced reader code is very large and/or complex
- native libraries or closed-source dependencies are required
- a public format specification document or reference implementation is not available
- the reader covers an imaging domain which is unfamiliar to the OME team
- test data cannot be provided to the OME team
- the reader needs to be updated outside of the normal Bio-Formats release process
- the reader code does not comply with OME code contribution policies
- developers/maintainers of the reader are not able to address feedback provided by the OME team during code review and testing

When considering a new reader, we suggest discussing with the OME team first before making substantial development. The OME team is responsible for making the final determination of whether a new reader should be defined as external.

How to propose an external reader

As indicated above, we suggest discussing with the OME team before investing significant time in a new reader.

At minimum, open a new pull request or issue on *ome/bioformats* that proposes to update *readers.txt* as indicated above, including the following information:

- where to find the reader source code
- who is maintaining the external reader, and an acknowledgement that the OME team is not responsible for maintaining this reader
- where/how the reader will be released
- test data, if possible
- supporting documentation (issues, forum threads, etc.) indicating why this reader is necessary

We strongly prefer to be able to test a proposed external reader before approving it, but recognize this may not always be possible. In these cases, we may ask for additional information in place of our own testing.

See also

OME's Third Party Contribution and Interaction Policy

What external reader ownership involves

Developing an external reader means taking complete ownership and responsibility for the development, maintenance, testing, and distribution of the reader and its dependencies. The OME team will not review code, fix issues, or otherwise participate in the development or release process. As a result, it is very important to have a maintenance and testing plan that includes:

- where issues should be reported
- who is responsible for fixing issues and releasing new versions
- where/how to find new versions of the reader, e.g. a Fiji update site

We strongly encourage reader maintainers to follow a support pattern similar to that used for core Bio-Formats, which includes:

- publicly accessible issue tracking
- participation in discussions on forum.image.sc
- public roadmaps and release announcements

Recategorizing an existing reader

If the OME team determines that an existing core reader meets the external reader criteria in *When to use an external reader*, and a suitable maintainer can be identified, then we may propose to recategorize the reader as external. This is our preferred alternative to removing it from both core Bio-Formats and readers.txt.

In the event that recategorizing is necessary, we will make contact with the proposed maintainer to discuss next steps.

3.4.6 Adding format/reader documentation pages

Documentation pages for the supported formats and readers are auto-generated during the build. This page explains how to amend/extend this part of the Bio-Formats documentation.

Formats

The supported formats pages are generated as part of the documentation build using **maven**. The metadata for each format is contained in format-pages.txt and the build generates a reStructuredText file for each format as well as an index page for all supported formats using Velocity.

The `format-pages.txt` is an INI file where each section corresponds to a particular format given by the section header. Multiple key/values should be defined for each section:

```
pagename
  The name of the output reStructuredText file. If unspecified, the section
  header will be used to generate the filename.

extensions
  The list of extensions supported for the format

owner
  The owner of the file format
```

(continues on next page)

(continued from previous page)

```
developer
  The developer of the file format

bsd
  A `yes/no` flag specifying whether the format readers/writers are under
  the BSD license

versions
  A comma-separated list of all versions supported for this format

weHave
  A bullet-point list describing the supporting material we have for this
  format including specification and sample datasets

weWant
  A bullet-point list describing the supporting material we would like to
  have for this format

pixelRating
metadataRating
opennessRating
presenceRating
utilityRating
  See :term:`Ratings legend and definitions`. Available choices are:
  `Poor`, `Fair`, `Good`, `Very Good`, `Outstanding`.
  The `metadataRating` should be set as follows:

    * base `metadataRating` is the smaller of `opennessRating` and `pixelsRating`
    * increment `metadataRating` by 1 if any combination of `Instrument.ID` plus `Image.
↪InstrumentRef`,
      or `Channel.EmissionWavelength`, or `Channel.ExcitationWavelength` are supported
    * increment `metadataRating` by 1 if any SPW metadata is supported or␣
↪:term:`pyramid` is ``yes``

reader
  A string or a comma-separated list of all readers for this format

mif
  set to ``true`` if the format can store multiple Images (in OME-XML
  terminology) or series (in Bio-Formats API terminology)

pyramid
  set to ``yes`` if the format can store a single image at multiple
  resolutions

notes
  Additional relevant information e.g. that we cannot distribute
  specification documents to third parties

options
  A link to additional reader and writer options documentation where they
  are available for the format
```

Dataset structure table

The summary table listing the extensions for each reader is generated by the build process reading their extensions and descriptions from all Bio-Formats readers (BSD and GPL). A reStructuredText file with a table summary of all file extensions is created.

Readers

The metadata pages for each reader are generated by the build process parsing their metadata support from all Bio-Formats readers (BSD and GPL). An intermediate `meta-support.txt` file is created which is then converted into one reStructuredText page for each reader as well as a metadata summary reStructuredText file using Velocity.

3.4.7 Bio-Formats service and dependency infrastructure

Description

The Bio-Formats service infrastructure is an interface driven pattern for dealing with external and internal dependencies. The design goal was mainly to avoid the cumbersome usage of `ReflectedUniverse` where possible and to clearly define both service dependency and interface between components. This is generally referred to as dependency injection, dependency inversion or component based design.

It was decided, at this point, to forgo the usage of potentially more powerful but also more complicated solutions such as:

- Spring
- Guice
- ...

The Wikipedia page for dependency injection contains many other implementations in many languages.

An added benefit is the potential code reuse possibilities as a result of decoupling of dependency and usage in Bio-Formats readers. Implementations of the initial Bio-Formats services were completed as part of BioFormatsCleanup and tickets #463 and #464.

Writing a service

- **Interface** – The basic form of a service is an interface which inherits from loci.common.services.Service. Here is a very basic example using the (now removed) `OMENotesService`

```
public interface OMENotesService extends Service {

  /**
   * Creates a new OME Notes instance.
   * @param filename Path to the file to create a Notes instance for.
   */
  public void newNotes(String filename);

}
```

- **Implementation** – This service then has an implementation, which is usually located in the Bio-Formats component or package which imports classes from an external, dynamic or other dependency. Again looking at the `OMENotesService`:

```
public class OMENotesServiceImpl extends AbstractService
  implements OMENotesService {
```

(continues on next page)

(continued from previous page)

```
  /**
   * Default constructor.
   */
  public OMENotesServiceImpl() {
    checkClassDependency(Notes.class);
  }

  /* (non-Javadoc)
   * @see loci.formats.dependency.OMENotesService#newNotes()
   */
  public void newNotes(String filename) {
    new Notes(null, filename);
  }

}
```

- **Style**
  - Extension of `AbstractService` to enable uniform runtime dependency checking is recommended. Java does not check class dependencies until classes are first instantiated so if you do not do this, you may end up with `ClassNotFound` or the like exceptions being emitted from your service methods. This is to be **strongly** discouraged. If a service has unresolvable classes on its CLASSPATH instantiation should fail, not service method invocation.
  - Service methods should not burden the implementer with numerous checked exceptions. Also external dependency exception instances should not be allowed to directly leak from a service interface. Please wrap these using a `ServiceException`.
  - By convention both the interface and implementation are expected to be in a package named `loci.*.services`. This is not a hard requirement but should be followed where possible.
- **Registration** – A service's interface and implementation must finally be *registered* with the loci.common.services.ServiceFactory via the `services.properties` file. Following the `OMENotesService` again, here is an example registration:

```
...
# OME notes service (implementation in legacy ome-notes component)
loci.common.services.OMENotesService=loci.ome.notes.services.OMENotesServiceImpl
...
```

Using a service

```
OMENotesService service = null;
try {
  ServiceFactory factory = new ServiceFactory();
  service = factory.getInstance(OMENotesService.class);
}
catch (DependencyException de) {
  LOGGER.info("", de);
}
...
```

3.4.8 Scripts for performing development tasks

The `tools` directory contains several scripts which are useful for building and performing routine updates to the code base.

test-build

This is the script used by Travis to test each commit. It compiles and runs tests on each of the components in the Bio-Formats repository according to the arguments specified. Valid arguments are:

- *clean*: cleans the Maven build directories
- *maven*: builds all Java components using Maven and runs unit tests
- *ant*: builds all Java components using Ant and runs unit tests
- *all*: equivalent of *clean maven ant*

update_copyright

This updates the end year in the copyright blocks of all source code files. The command takes no arguments, and sets the end year to be the current year. As *update_copyright* is a Bash script, it is not intended to be run on Windows.

For more general guidance about how to contribute to OME projects, see the Contributing developers documentation.

3.5 Issue tracking

See the Bio-Formats repository on GitHub for information on work currently planned or in progress.

CHAPTER

FOUR

FORMATS

Bio-Formats supports reading over 140 different file formats, and writing several common open formats. The *Dataset Structure Table* explains the file extension you should choose to open/import a dataset in any of these formats, while the *Supported Formats* table lists all of the formats and gives an indication of how well they are supported and whether Bio-Formats can write, as well as read, each format. The *Summary of supported metadata fields* table shows an overview of the *OME data model* fields populated for each format.

Further information on each individual format is also provided (click on the entries in the supported formats table).

We are always looking for examples of files to help us provide better support for different formats. If you would like to help, you can upload files to the Bio-Formats Zenodo community and let us know. If you have any questions, or would prefer not to use Zenodo, please contact us via the Image.sc forum. If your format is already supported, please refer to the 'we would like to have' section on the individual page for that format, to see if your dataset would be useful to us.

All the example files we have permission to share publicly are freely available from our sample image downloads site.

4.1 Dataset Structure Table

This table shows the extension of the file that you should choose if you want to open/import a dataset in a particular format.

Format name	File to choose	Structure of files
AIM	.aim	Single file
ARF	.arf	Single file
Adobe Photoshop	.psd	Single file
Adobe Photoshop TIFF	.tif, .tiff	Single file
Alicona AL3D	.al3d	Single file
Amersham Biosciences GEL	.gel	Single file
Amira	.am, .amiramesh, .grey, .hx, .labels	Single file
Analyze 7.5	.img, .hdr	One .img file and one similarly-named .hdr file
Andor SIF	.sif	Single file
Animated PNG	.png	Single file
Aperio AFI	.afi	One .afi file and several similarly-named .svs files
Aperio SVS	.svs	Single file
Audio Video Interleave	.avi	Single file
BD Pathway	.exp, .tif	Multiple files (.exp, .dye, .ltp, ...) plus one or more directories containing .tif and .bmp files

continues on next page

Table 1 – continued from previous page

Format name	File to choose	Structure of files
BDV	.xml, .h5	Single file
Bio-Rad GEL	.1sc	Single file
Bio-Rad PIC	.pic, .xml, .raw	One or more .pic files and an optional lse.xml file
Bio-Rad SCN	.scn	Single file
Bitplane Imaris	.ims	Single file
Bitplane Imaris 3 (TIFF)	.ims	Single file
Bitplane Imaris 5.5 (HDF)	.ims	Single file
Bruker	(no extension)	One 'fid' and one 'acqp' plus several other metadata files and a 'pdata' directory
Burleigh	.img	Single file
Canon RAW	.cr2, .crw, .jpg, .thm, .wav	Single file
CellH5 (HDF)	.ch5	Single file
CellSens VSI	.vsi, .ets	One .vsi file and an optional directory with a similar name that contains at least one subdirectory with .ets files
CellVoyager	.tif, .xml	Directory with 2 master files 'MeasurementResult.xml' and 'MeasurementResult.ome.xml', used to stitch together several TIF files.
CellWorx	.pnl, .htd, .log	One .htd file plus one or more .pnl files and optionally one or more .log files
Cellomics C01	.c01, .dib	One or more .c01 files
Compix Simple-PCI	.cxd	Single file
DICOM	.dic, .dcm, .dicom, .jp2, .j2ki, .j2kr, .raw, .ima	One or more .dcm or .dicom files
DNG	.cr2, .crw, .jpg, .thm, .wav, .tif, .tiff	Single file
Deltavision	.dv, .r3d, .r3d_d3d, .dv.log, .r3d.log	One .dv, .r3d, or .d3d file and up to two optional .log files
ECAT7	.v	Single file
Encapsulated PostScript	.eps, .epsi, .ps	Single file
Evotec Flex	.flex, .mea, .res	One directory containing one or more .flex files, and an optional directory containing an .mea and .res file. The .mea and .res files may also be in the same directory as the .flex file(s).
Extended leica file	.xlef	Single file
FEI TIFF	.tif, .tiff	Single file
FEI/Philips	.img	Single file
Flexible Image Transport System	.fits, .fts	Single file
FlowSight	.cif	Single file
Fuji LAS 3000	.img, .inf	Single file
Gatan DM2	.dm2	Single file
Gatan Digital Micrograph	.dm3, .dm4	Single file
Graphics Interchange Format	.gif	Single file
Hamamatsu Aquacosmos	.naf	Single file

continues on next page

Table 1 – continued from previous page

Format name	File to choose	Structure of files
Hamamatsu DCIMG	.dcimg	Single file
Hamamatsu HIS	.his	Single file
Hamamatsu NDPI	.ndpi	Single file
Hamamatsu NDPIS	.ndpis	One .ndpis file and at least one .ndpi file
Hamamatsu VMS	.vms	One .vms file plus several .jpg files
Hitachi	.txt	One .txt file plus one similarly-named .tif, .bmp, or .jpg file
I2I	.i2i	Single file
IMAGIC	.hed, .img	One .hed file plus one similarly-named .img file
IMOD	.mod	Single file
INR	.inr	Single file
IPLab	.ipl	Single file
IVision	.ipm	Single file
Imacon	.fff	Single file
Image Cytometry Standard	.ics, .ids	One .ics and possibly one .ids with a similar name
Image-Pro Sequence	.seq, .ips	Single file
Image-Pro Workspace	.ipw	Single file
Improvision TIFF	.tif, .tiff	Single file
InCell 1000/2000	.xdce, .xml, .tiff, .tif, .xlog	One .xdce file with at least one .tif/.tiff or .im file
InCell 3000	.frm	Single file
Inveon	.hdr	One .hdr file plus one similarly-named file
Ionpath MIBI	.tif, tiff	Single file
JEOL	.dat, .img, .par	A single .dat file or an .img file with a similarly-named .par file
JPEG	.jpg, .jpeg, .jpe	Single file
JPEG-2000	.jp2, .j2k, .jpf	Single file
JPK Instruments	.jpk	Single file
JPX	.jpx	Single file
KLB	.klb	Single file
Khoros XV	.xv	Single file
Kodak Molecular Imaging	.bip	Single file
LEO	.sxm, .tif, .tiff	Single file
LI-FLIM	.fli	Single file
Laboratory Imaging	.lim	Single file
Lavision Imspector	.msr	Single file
Leica	.lei, .tif, .tiff, .raw	One .lei file with at least one .tif/.tiff file and an optional .txt file
Leica Image File Format	.lif	Single file
Leica Object Format	.lof	Single file
Leica SCN	.scn	Single file
Leica TCS TIFF	.tif, .tiff, .xml	Single file
Li-Cor L2D	.l2d, .scn, .tif	One .l2d file with one or more directories containing .tif/.tiff files

continues on next page

Table 1 – continued from previous page

Format name	File to choose	Structure of files
MIAS	.tif, .tiff, .txt	One directory per plate containing one directory per well, each with one or more .tif/.tiff files
MINC MRI	.mnc	Single file
Medical Research Council	.mrc, .st, .ali, .map, .rec, .mrcs	Single file
MetaXpress TIFF	.htd, .tif	One .htd file plus one or more .tif files
Metamorph STK	.stk, .nd, .scan, .tif, .tiff	One or more .stk or .tif/.tiff files plus an optional .nd or .scan file
Metamorph TIFF	.tif, .tiff	One or more .tif/.tiff files
Micro-Manager	.tif, .tiff, .txt, .xml	A file ending in 'metadata.txt' plus one or more .tif files
MicroCT	.vff	Directory with XML file and one .tif/.tiff file per plane
Mikroscan TIFF	.tif, .tiff	Single file
Minolta MRW	.mrw	Single file
Molecular Devices JDCE	.jdce	One .jdce (JSON) file with at least one .tif/.tiff file
Molecular Imaging	.stp	Single file
Multiple-image Network Graphics	.mng	Single file
NIfTI	.nii, .img, .hdr, .nii.gz	A single .nii file or a single .nii.gz file or one .img file and a similarly-named .hdr file
NOAA-HRD Gridded Data Format	(no extension)	Single file
NRRD	.nrrd, .nhdr	A single .nrrd file or one .nhdr file and one other file containing the pixels
Nikon Elements TIFF	.tif, .tiff	Single file
Nikon ND2	.nd2, .jp2	Single file
Nikon NEF	.nef, .tif, .tiff	Single file
Nikon TIFF	.tif, .tiff	Single file
OBF	.obf, .msr	OBF file
OME-TIFF	.ome.tiff, .ome.tif, .ome.tf2, .ome.tf8, .ome.btf, .companion.ome	One or more .ome.tiff files
OME-XML	.ome, .ome.xml	Single file
Olympus .omp2info	.omp2info	One .omp2info file and at least one .oir or .vsi file
Olympus APL	.apl, .tnb, .mtb, .tif	One .apl file, one .mtb file, one .tnb file, and a directory containing one or more .tif files
Olympus FV1000	.oib, .oif, .pty, .lut	Single .oib file or one .oif file and a similarly-named directory containing .tif/.tiff files
Olympus Fluoview/ABD TIFF	.tif, .tiff	One or more .tif/.tiff files, and an optional .txt file
Olympus OIR	.oir	Single file
Olympus SIS TIFF	.tif, .tiff	Single file
Olympus ScanR	.dat, .xml, .tif	One .xml file, one 'data' directory containing .tif/.tiff files, and optionally two .dat files

continues on next page

Table 1 – continued from previous page

Format name	File to choose	Structure of files
Olympus Slidebook	.sld, .spl	Single file
Openlab LIFF	.liff	Single file
Openlab RAW	.raw	Single file
Oxford Instruments	.top	Single file
PCO-RAW	.pcoraw, .rec	A single .pcoraw file with a similarly-named .rec file
PCX	.pcx	Single file
PICT	.pict, .pct	Single file
POV-Ray	.df3	Single file
Perkin Elmer Densitometer	.hdr, .img	One .hdr file and a similarly-named .img file
Perkin-Elmer Nuance IM3	.im3	Single file
PerkinElmer	.ano, .cfg, .csv, .htm, .rec, .tim, .zpo, .tif	One .htm file, several other metadata files (.tim, .ano, .csv, ...) and either .tif files or .2, .3, .4, etc. files
PerkinElmer Columbus	.xml	Directory with XML file and one .tif/.tiff file per plane
PerkinElmer Operetta	.tif, .tiff, .xml	Directory with XML file and one .tif/.tiff file per plane
PerkinElmer Vectra/QPTIFF	.tiff, .tif, .qptiff	Single file
PicoQuant Bin	.bin	Single file
Portable Any Map	.pbm, .pgm, .ppm	Single file
Prairie TIFF	.tif, .tiff, .cfg, .env, .xml	One .xml file, one .cfg file, and one or more .tif/.tiff files
Princeton Instruments SPE	.spe	Single file
Pyramid TIFF	.tif, .tiff	Single file
Quesant AFM	.afm	Single file
QuickTime	.mov	Single file
RCPNL	.rcpnl	One .dv, .r3d, or .d3d file and up to two optional .log files
RHK Technologies	.sm2, .sm3	Single file
SBIG	(no extension)	Single file
SM Camera	(no extension)	Single file
SPC FIFO Data	.spc, .set	One .spc file and similarly named .set file
SPCImage Data	.sdt	Single file
SPIDER	.spi	Single file
Seiko	.xqd, .xqf	Single file
SimplePCI TIFF	.tif, .tiff	Single file
Simulated data	.fake	Single file
SlideBook 7 SLD (native)	.sldy, .sldyz	Single file
Slidebook TIFF	.tif, .tiff	Single file
Tagged Image File Format	.tif, .tiff, .tf2, .tf8, .btf	Single file
Tecan Spark Cyto	.db	SQLite database, TIFF files, optional analysis output
Text	.txt, .csv	Single file
TillVision	.vws, .pst, .inf	One .vws file and possibly one similarly-named directory
TissueFAXS	.aqproj, .tfcyto	An .aqproj file with one or more .tfcyto database files

continues on next page

Table 1 – continued from previous page

Format name	File to choose	Structure of files
TopoMetrix	.tfr, .ffr, .zfr, .zfp, .2fl	Single file
Trestle	.tif	One .tif file plus several other similarly-named files (e.g. *.FocalPlane-*, .sld, .slx, .ROI)
Truevision Targa	.tga	Single file
UBM	.pr3	Single file
Unisoku STM	.hdr, .dat	One .HDR file plus one similarly-named .DAT file
VG SAM	.dti	Single file
Varian FDF	.fdf	Single file
Veeco	.hdf	Single file
Ventana .bif	.bif	Single file
Visitech XYS	.xys, .html	One .html file plus one or more .xys files
Volocity Library	.mvd2, .aisf, .aiix, .dat, .atsf	One .mvd2 file plus a 'Data' directory
Volocity Library Clipping	.acff	Single file
WA Technology TOP	.wat	Single file
Windows Bitmap	.bmp	Single file
Yokogawa CV7000	.wpi	Directory with XML files and one .tif/.tiff file per plane
Zeiss AxioVision TIFF	.tif, .xml	Single file
Zeiss CZI	.czi	Single file
Zeiss LMS	.lms	Single file
Zeiss Laser-Scanning Microscopy	.lsm, .mdb	One or more .lsm files; if multiple .lsm files are present, an .mdb file should also be present
Zeiss Vision Image (ZVI)	.zvi	Single file
Zeiss XRM	.txm, .txrm	Single file
Zip	.zip	Single file

4.1.1 Flex Support

OMERO.importer supports importing analyzed Flex files from an Opera system.

Basic configuration is done via the `importer.ini`. Once the user has run the Importer once, this file will be in the following location:

- `C:\Documents and Settings\<username>\omero\importer.ini`

The user will need to modify or add the `[FlexReaderServerMaps]` section of the INI file as follows:

```
...
[FlexReaderServerMaps]
CIA-1 = \\\\hostname1\\mount;\\\\archivehost1\\mount
CIA-2 = \\\\hostname2\\mount;\\\\archivehost2\\mount
```

where the *key* of the INI file line is the value of the "Host" tag in the `.mea` measurement XML file (here: `<Host name="CIA-1">`) and the value is a semicolon-separated list of *escaped* UNC path names to the Opera workstations where the Flex files reside.

Once this resolution has been encoded in the configuration file **and** you have restarted the importer, you will be able to select the `.mea` measurement XML file from the Importer user interface as the import target.

4.2 Supported Formats

Ratings legend and definitions

Format	Extensions	Pixels	Metadata	Openness	Presence	Utility	Export	BSD	Multiple Images	Pyramid
3i SlideBook	.sld	▲	▭	▼	▲	▼	✗	✗	✓	✗
3i SlideBook 7	.sldy	▲▲	▲▲	▲	▲	▲	✗	✓	✗	✗
Andor Bio-Imaging Division (ABD) TIFF	.tif	▲	▲	▭	▼	▭	✗	✗	✓	✗
AIM	.aim	▭	▼	▼	▼▼	▼	✗	✗	✗	✗
Alicona 3D	.al3d	▲	▲	▲	▼▼	▭	✗	✗	✗	✗
Amersham Biosciences Gel	.gel	▲	▭	▭	▼	▼	✗	✗	✗	✗
Amira Mesh	.am, .amiramesh, .grey, .hx, .labels	▲	▼	▼	▼	▼	✗	✗	✗	✗
Amnis FlowSight	.cif	▭	▭	▭	▼▼	▼	✗	✓	✓	✗
Analyze 7.5	.img, .hdr	▲	▲	▲	▭	▼	✗	✗	✗	✗
Andor SIF	.sif	▭	▼	▼	▼	▼	✗	✗	✗	✗
Animated PNG	.png	▲	▲	▲▲	▭	▼▼	✓	✓	✗	✗
Aperio AFI	.afi, .svs	▲	▲	▲	▭	▭	✗	✗	✓	✓
Aperio SVS TIFF	.svs	▲	▲	▲	▭	▭	✗	✗	✓	✓
Applied Precision CellWorX	.htd, .pnl	▲	▲	▭	▼	▼	✗	✗	✓	✗
AVI (Audio Video Interleave)	.avi	▭	▼	▼	▲▲	▼▼	✓	✓	✗	✗
Axon Raw Format	.arf	▲	▲	▲	▼▼	▼	✗	✗	✗	✗
BD Pathway	.exp, .tif	▲	▲	▭	▼	▭	✗	✗	✓	✗
Becker & Hickl SPC FIFO	.spc	▼▼	▼▼	▭	▼▼	▼	✗	✗	✓	✗
Becker & Hickl SPCImage	.sdt	▲	▭	▭	▼▼	▼	✗	✗	✓	✗
Big Data Viewer	.xml, .h5	▭	▲	▲	▭	▭	✗	✓	✓	✓
Bio-Rad Gel	.1sc	▭	▼	▼	▼▼	▼	✗	✗	✗	✗
Bio-Rad PIC	.pic, .raw, .xml	▲▲	▲▲	▲	▲	▲	✗	✗	✗	✗
Bio-Rad SCN	.scn	▲	▼	▼	▼▼	▼	✗	✗	✗	✗
Bitplane Imaris	.ims	▲	▲	▲	▼	▼	✗	✗	✓	✓
Bruker MRI		▭	▼	▼	▭	▼	✗	✗	✓	✗
Burleigh	.img	▭	▼	▼	▼	▼	✗	✗	✗	✗

continues on next page

Table 2 – continued from previous page

Format	Extensions	Pixels	Metadata	Openness	Presence	Utility	Export	BSD	Multiple Images	Pyramid
Canon DNG	.cr2, .crw									
CellH5	.ch5									
Cellomics	.c01, .dib									
cellSens VSI	.vsi									
CellVoyager	.xml, .tif									
CV7000	.wpi, .tif									
DeltaVision	.dv, .r3d, .rcpnl									
DICOM	.dcm, .dicom									
ECAT7	.v									
EPS (Encapsulated PostScript)	.eps, .epsi, .ps									
Evotec/PerkinElmer Opera Flex	.flex, .mea, .res									
FEI	.img									
FEI TIFF	.tiff									
FITS (Flexible Image Transport System)	.fits									
Gatan Digital Micrograph	.dm3, .dm4									
Gatan Digital Micrograph 2	.dm2									
GE MicroCT	.vff									
GIF (Graphics Interchange Format)	.gif									
Hamamatsu Aquacosmos NAF	.naf									
Hamamatsu DCIMG	.dcimg									
Hamamatsu HIS	.his									
Hamamatsu ndpi	.ndpi, .ndpis									
Hamamatsu VMS	.vms									
Hitachi S-4800	.txt, .tif, .bmp, .jpg									
I2I	.i2i									
ICS (Image Cytometry Standard)	.ics, .ids									
Imacon	.fff									

continues on next page

Table 2 – continued from previous page

Format	Extensions	Pixels	Metadata	Openness	Presence	Utility	Export	BSD	Multiple Images	Pyramid
ImagePro Sequence	.seq									
ImagePro Workspace	.ipw									
IMAGIC	.hed, .img									
IMOD	.mod									
Improvision Openlab LIFF	.liff									
Improvision Openlab Raw	.raw									
Improvision TIFF	.tif									
Imspector OBF	.obf, .msr									
InCell 1000/2000	.xdce, .tif									
InCell 3000	.frm									
INR	.inr									
Inveon	.hdr									
Ionpath MIBI	.tif, .tiff									
IPLab	.ipl									
IVision	.ipm									
JDCE	.jdce, .tif									
JEOL	.dat, .img, .par									
JPEG	.jpg									
JPEG 2000	.jp2, .j2k, .jpf									
JPK	.jpk									
JPX	.jpx									
Keller Lab Block	.klb									
Khoros VIFF (Visualization Image File Format) Bitmap	.xv									
Kodak BIP	.bip									
Lambert Instruments FLIM	.fli									
LaVision Imspector	.msr									
Leica LCS LEI	.lei, .tif									
Leica LAS AF LIF (Leica Image File Format)	.lif									
Leica LOF	.lof									
Leica SCN	.scn									

continues on next page

Table 2 – continued from previous page

Format	Extensions	Pixels	Metadata	Openness	Presence	Utility	Export	BSD	Multiple Images	Pyramid
Leica XLEF	.xlef									
LEO	.sxm, .tif, .tiff									
Li-Cor L2D	.l2d, .tif, .scn									
LIM (Laboratory Imaging/Nikon)	.lim									
MetaMorph 7.5 TIFF	.tiff									
MetaMorph Stack (STK)	.stk, .nd									
MetaXpress	.htd, .tif, .tiff									
MIAS (Maia Scientific)	.tif									
Micro-Manager	.tif, .txt, .xml									
Mikroscan TIFF	.tif, .tiff									
MINC MRI	.mnc									
Minolta MRW	.mrw									
MNG (Multiple-image Network Graphics)	.mng									
Molecular Imaging	.stp									
MRC (Medical Research Council)	.mrc, .st, .ali, .map, .rec, .mrcs									
NEF (Nikon Electronic Format)	.nef, .tif									
NIfTI	.img, .hdr, .nii, .nii.gz									
Nikon Elements TIFF	.tiff									
Nikon EZ-C1 TIFF	.tiff									
Nikon NIS-Elements ND2	.nd2									
NRRD (Nearly Raw Raster Data)	.nrrd, .nhdr, .raw, .txt									
Olympus CellR/APL	.apl, .mtb, .tnb, .tif, .obsep									
Olympus FluoView FV1000	.oib, .oif									
Olympus FluoView TIFF	.tif									
Olympus OIR	.oir									
Olympus OMP2INFO	.omp2info									
Olympus ScanR	.xml, .dat, .tif									

continues on next page

Table 2 – continued from previous page

Format	Extensions	Pixels	Metadata	Openness	Presence	Utility	Export	BSD	Multiple Images	Pyramid
Olympus SIS TIFF	.tiff						✗	✗	✗	✗
OME-TIFF	.ome.tiff, .ome.tif, .ome.tf2, .ome.tf8, .ome.btf						✓	✓	✓	✓
OME-XML	.ome, .ome.xml						✓	✓	✓	✗
OMERO Pyramid							✗	✓	✓	✓
Oxford Instruments	.top						✗	✗	✗	✗
PCORAW	.pcoraw, .rec						✗	✗	✗	✗
PCX (PC Paintbrush)	.pcx						✗	✓	✗	✗
Perkin Elmer Densitometer	.pds						✗	✗	✗	✗
PerkinElmer Columbus	.xml, .csv, .tif						✗	✗	✓	✗
PerkinElmer Nuance	.im3						✗	✓	✓	✗
PerkinElmer Operetta	.tiff, .xml						✗	✗	✓	✗
PerkinElmer UltraVIEW	.tif, .2, .3, .4, etc.						✗	✗	✗	✗
Portable Any Map	.pbm, .pgm, .ppm						✗	✓	✗	✗
Adobe Photoshop PSD	.psd						✗	✗	✗	✗
Photoshop TIFF	.tif, .tiff						✗	✗	✓	✗
PicoQuant Bin	.bin						✗	✗	✗	✗
PICT (Macintosh Picture)	.pict						✗	✓	✗	✗
PNG (Portable Network Graphics)	.png						✓	✓	✗	✗
Prairie Technologies TIFF	.tif, .xml, .cfg						✗	✗	✓	✗
Princeton Instruments SPE	.spe						✗	✗	✓	✗
Quesant	.afm						✗	✗	✗	✗
QuickTime Movie	.mov						✓	✓	✗	✗
RHK	.sm2, .sm3						✗	✗	✗	✗
SBIG							✗	✗	✗	✗
Seiko	.xqd, .xqf						✗	✗	✗	✗
SimplePCI & HCImage	.cxd						✗	✗	✗	✗

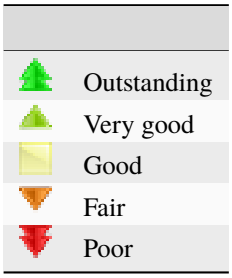
continues on next page

Table 2 – continued from previous page

Format	Extensions	Pixels	Metadata	Openness	Presence	Utility	Export	BSD	Multiple Images	Pyramid
SimplePCI & HCImage TIFF	.tiff									
SM Camera										
SPIDER	.spi, .stk									
Targa	.tga									
Tecan Spark Cyto Workspace	.db, .tif									
Text	.txt									
TIFF (Tagged Image File Format)	.tiff, .tif, .tf2, .tf8, .btf									
TillPhotonics TillVision	.vws									
TissueGnostics TFCYTO	.tfcyto, .aqproj									
Topometrix	.tfr, .ffr, .zfr, .zfp, .2fl									
Trestle	.tif, .sld, .jpg									
UBM	.pr3									
Unisoku	.dat, .hdr									
Varian FDF	.fdf									
Vectra QPTIFF	.tif, .qptiff									
Veeco AFM	.hdf									
Ventana BIF	.bif									
VG SAM	.dti									
VisiTech XYS	.xys, .html									
Volocity	.mvd2									
Volocity Library Clipping	.acff									
WA-TOP	.wat									
Windows Bitmap	.bmp									
Zeiss Axio CSM	.lms									
Zeiss AxioVision TIFF	.xml, .tif									
Zeiss AxioVision ZVI (Zeiss Vision Image)	.zvi									
Zeiss CZI	.czi									
Zeiss LSM (Laser Scanning Microscope) 510/710	.lsm, .mdb									
Zeiss XRM	.txm, .txrm									

Bio-Formats currently supports **165** formats

Ratings legend and definitions



Pixels
: Our estimation of Bio-Formats' ability to reliably extract complete and accurate pixel values from files in that format. The better this score, the more confident we are that Bio-Formats will successfully read your file without displaying an error message or displaying an erroneous image.

Metadata
: Our certainty in the thoroughness and correctness of Bio-Formats' metadata extraction and conversion from files of that format into standard OME-XML. The better this score, the more confident we are that all meaningful metadata will be parsed and populated as OME-XML.

Openness
: This is not a direct expression of Bio-Formats' performance, but rather indicates the level of cooperation the format's controlling interest has demonstrated toward the scientific community with respect to the format. The better this score, the more tools (specification documents, source code, sample files, etc.) have been made available.

Presence
: This is also not directly related to Bio-Formats, but instead represents our understanding of the format's popularity, and is also as a measure of compatibility between applications. The better this score, the more common the format and the more software packages include support for it.

Utility
: Our opinion of the format's suitability for storing metadata-rich microscopy image data. The better this score, the wider the variety of information that can be effectively stored in the format.

Export
: This indicates whether Bio-Formats is capable of writing the format (Bio-Formats can read every format on this list).

BSD
: This indicates whether format is BSD-licensed. By default, format readers and writers are GPL-licensed.

Multiple Images
: This indicates whether the format can store multiple Images (in OME-XML terminology) or series (in Bio-Formats API terminology).

Pyramid
: This indicates whether the format can store a single image at multiple resolutions, typically referred to as an image pyramid.

4.2.1 3i SlideBook

Extensions: .sld

Developer: Intelligent Imaging Innovations

Owner: Intelligent Imaging Innovations

Support

BSD-licensed:

Export:

Officially Supported Versions: 4.1, 4.2, 5.0, 5.5, 6.0

Reader: SlidebookReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Numerous SlideBook datasets

We would like to have:

- A SlideBook specification document
- More SlideBook datasets (preferably acquired with the most recent SlideBook software)

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

We strongly encourage users to export their .sld files to OME-TIFF using the SlideBook software. Bio-Formats is not likely to support the full range of metadata that is included in .sld files, and so exporting to OME-TIFF from SlideBook is the best way to ensure that all metadata is preserved. Free software from 3i can export the files to OME-TIFF post-acquisition, see https://www.intelligent-imaging.com/slidebook.

3i also develops a native SlideBook reader which works with Bio-Formats. See http://www.openmicroscopy.org/info/slidebook for details.

See also

Slidebook software overview

4.2.2 3i SlideBook 7

Extensions: .sldy

Developer: Intelligent Imaging Innovations

Owner: Intelligent Imaging Innovations

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SlideBook7Reader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- SlideBook

We currently have:

- a small number of datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.3 Andor Bio-Imaging Division (ABD) TIFF

Extensions: .tif

Developer: Andor Bioimaging Department

Owner: Andor Technology

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: FluoviewReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an ABD-TIFF specification document (from 2005 November, in PDF)
- a few ABD-TIFF datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

With a few minor exceptions, the ABD-TIFF format is identical to the Fluoview TIFF format.

4.2.4 AIM

Extensions: .aim

Developer: SCANCO Medical AG

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: AIMReader (Source Code, *Supported Metadata Fields*)

We currently have:

- one .aim file

We would like to have:

- an .aim specification document
- more .aim files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.5 Alicona 3D

Extensions: .al3d

Owner: Alicona Imaging

Support

BSD-licensed:

Export:

Officially Supported Versions: 1.0

Reader: AliconaReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an AL3D specification document (v1.0, from 2003, in PDF)

- a few AL3D datasets

We would like to have:

- more AL3D datasets (Z series, T series, 16-bit)

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Known deficiencies:

- Support for 16-bit AL3D images is present, but has never been tested.
- Texture data is currently ignored.

4.2.6 Amersham Biosciences Gel

Extensions: .gel

Developer: Molecular Dynamics

Owner: Cytiva

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: GelReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a GEL specification document (Revision 2, from 2001 Mar 15, in PDF)
- a few GEL datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

See also

GEL Technical Overview

4.2.7 Amira Mesh

Extensions: .am, .amiramesh, .grey, .hx, .labels

Developer: Visage Imaging

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: AmiraReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few Amira Mesh datasets
- public sample images

We would like to have:

- more Amira Mesh datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.8 Amnis FlowSight

Extensions: .cif

Owner: Amnis (now owned by Merck)

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: FlowSightReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few sample datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.9 Analyze 7.5

Extensions: .img, .hdr

Developer: Mayo Foundation Biomedical Imaging Resource

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: AnalyzeReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an Analyze 7.5 specification document
- several Analyze 7.5 datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.10 Andor SIF

Extensions: .sif

Developer: Andor Bioimaging Department

Owner: Andor Technology

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SIFReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a small number of Andor SIF datasets

We would like to have:

- an Andor SIF specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.11 Animated PNG

Extensions: .png

Developer: The Animated PNG Project

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: APNGReader (Source Code, *Supported Metadata Fields*)

Writer: APNGWriter (Source Code)

Freely Available Software:

- Firefox 3+
- Opera 9.5+
- KSquirrel

We currently have:

- a specification document
- several APNG files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.12 Aperio AFI

Extensions: .afi, .svs

Owner: Leica Biosystems

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: AFIReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several AFI datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

> **See also**
>
> Aperio ImageScope

4.2.13 Aperio SVS TIFF

Extensions: .svs

Owner: Leica Biosystems

Support

BSD-licensed:

Export:

Officially Supported Versions: 8.0, 8.2, 9.0

Reader: SVSReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- OpenSlide

We currently have:

- many SVS datasets
- public sample images
- an SVS specification document

- the ability to generate additional SVS datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

Additional options are available for reading or writing this format type, see *Additional reader and writer options* for information.

See also

Aperio ImageScope

4.2.14 Applied Precision CellWorX

Extensions: .htd, .pnl

Developer: Applied Precision, Inc.

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: CellWorxReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few CellWorX datasets
- public sample datasets

We would like to have:

- a CellWorX specification document
- more CellWorX datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.15 AVI (Audio Video Interleave)

Extensions: .avi

Developer: Microsoft

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: AVIReader (Source Code, *Supported Metadata Fields*)

Writer: AVIWriter (Source Code)

Freely Available Software:

- AVI Reader plugin for ImageJ
- AVI Writer plugin for ImageJ

We currently have:

- several AVI datasets

We would like to have:

- more AVI datasets, including:
  - files with audio tracks and/or multiple video tracks
  - files compressed with a common unsupported codec
  - 2+ GB files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

- Bio-Formats can save image stacks as AVI (uncompressed).
- The following codecs are supported for reading:
  - Microsoft Run-Length Encoding (MSRLE)
  - Microsoft Video (MSV1)
  - Raw (uncompressed)
  - JPEG

See also

AVI RIFF File Reference AVI on Wikipedia

4.2.16 Axon Raw Format

Extensions: .arf

Owner: INDEC BioSystems

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ARFReader (Source Code, *Supported Metadata Fields*)

We currently have:

- one ARF dataset
- a specification document

We would like to have:

- more ARF datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.17 BD Pathway

Extensions: .exp, .tif

Owner: BD Biosciences

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: BDReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few BD Pathway datasets

We would like to have:

- more BD Pathway datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.18 Becker & Hickl SPC FIFO

Extensions: .spc

Owner: Becker-Hickl

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SPCReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an SPC specification document
- public sample images

We would like to have:

- more SPC sample files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

- Only files containing frame, line and pixel clock information are currently supported

4.2.19 Becker & Hickl SPCImage

Extensions: .sdt

Owner: Becker-Hickl

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SDTReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an SDT specification document (from 2008 April, in PDF)
- an SDT specification document (from 2006 June, in PDF)
- Becker & Hickl's SPCImage software
- a large number of SDT datasets
- public sample images
- the ability to produce new datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.20 Big Data Viewer

Extensions: .xml, .h5

Owner: Tobias Pietzsch

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: BDVReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a BDV specification document
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.21 Bio-Rad Gel

Extensions: .1sc

Owner: Bio-Rad

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: BioRadGelReader (Source Code, *Supported Metadata Fields*)

We currently have:

- software that can read Bio-Rad Gel files
- several Bio-Rad Gel files
- reverse-engineered Bio-Rad Gel (1sc) file format specification
- Bio-Rad's Image Lab 5.2.1 software

We would like to have:

- more Bio-Rad Gel files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

- Bio-Rad Gel Format (1sc) specification
- Python3 package of a Bio-Rad Gel Format (1sc) reader
  - `pip3 install biorad1sc_reader`
  - biorad1sc_reader documentation
  - Includes command-line utilities
    - bio1sc2tiff - convert 1sc file image to tiff
    - bio1scmeta - report all metadata in a 1sc file
    - bio1scread - report details on internal 1sc file structure
  - biorad1sc_reader on pypi
  - biorad1sc_reader on github

4.2.22 Bio-Rad PIC

Extensions: .pic, .raw, .xml

Developer: Bio-Rad

Owner: ZEISS International

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: BioRadReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Bio-Rad PIC reader plugin for ImageJ

We currently have:

- a PIC specification document (v4.5, in PDF)
- an older PIC specification document (v4.2, from 1996 December 16, in DOC)
- a large number of PIC datasets
- the ability to produce new datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

- Commercial applications that support this format include:
  - Bitplane Imaris
  - SVI Huygens

4.2.23 Bio-Rad SCN

Extensions: .scn

Developer: Bio-Rad

Owner: Bio-Rad

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: BioRadSCNReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few Bio-Rad .scn files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.24 Bitplane Imaris

Extensions: .ims

Owner: Oxford Instruments (formerly Bitplane)

Support

BSD-licensed:

Export:

Officially Supported Versions: 2.7, 3.0, 5.5

Readers:

- ImarisHDFReader (Source Code, *Supported Metadata Fields*)
- ImarisTiffReader (Source Code, *Supported Metadata Fields*)
- ImarisReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Imaris writer

We currently have:

- an Imaris (RAW) specification document (from no later than 1997 November 11, in HTML)
- an Imaris 5.5 (HDF) specification document
- Bitplane's bfFileReaderImaris3N code (from no later than 2005, in C++)
- several older Imaris (RAW) datasets
- one Imaris 3 (TIFF) dataset
- several Imaris 5.5 (HDF) datasets
- public sample images

We would like to have:

- an Imaris 3 (TIFF) specification document
- more Imaris 3 (TIFF) datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

- **There are three distinct Imaris formats:**
    1. the old binary format (introduced in Imaris version 2.7)
    2. Imaris 3, a TIFF variant (introduced in Imaris version 3.0)
    3. Imaris 5.5, an HDF variant (introduced in Imaris version 5.5)

4.2.25 Bruker MRI

Developer: Bruker

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: BrukerReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Bruker plugin for ImageJ

We currently have:

- a few Bruker MRI datasets

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.26 Burleigh

Extensions: .img

Owner: Burleigh Instruments

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: BurleighReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read Burleigh files (from ImageSXM)
- a few Burleigh files

We would like to have:

- a Burleigh file format specification
- more Burleigh files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.27 Canon DNG

Extensions: .cr2, .crw

Developer: Canon

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: DNGReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- IrfanView

We currently have:

- a few example datasets

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.28 CellH5

Extensions: .ch5

Developer: CellH5

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: CellH5Reader (Source Code, *Supported Metadata Fields*)

Writer: CellH5Writer (Source Code)

Freely Available Software:

- CellH5

We currently have:

- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.29 Cellomics

Extensions: .c01, .dib

Developer: Thermo Fisher Scientific

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: CellomicsReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few Cellomics .c01 datasets
- public sample images

We would like to have:

- a Cellomics .c01 specification document
- more Cellomics .c01 datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.30 cellSens VSI

Extensions: .vsi

Developer: Olympus

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: CellSensReader (Source Code, *Supported Metadata Fields*)

We currently have:

- many example datasets
- public sample images
- a VSI specification document (v1.6, 2012 November 27, in PDF)
- a VSI specification document (v1.3, 2010 February 5, in PDF)

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.31 CellVoyager

Extensions: .xml, .tif

Owner: Yokogawa

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: CellVoyagerReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few example datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.32 CV7000

Extensions: .wpi, .tif

Owner: Yokogawa

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: CV7000Reader (Source Code, *Supported Metadata Fields*)

We currently have:

- many example datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Additional options are available for reading or writing this format type, see *Additional reader and writer options* for information.

4.2.33 DeltaVision

Extensions: .dv, .r3d, .rcpnl

Owner: Cytiva (formerly GE Healthcare, Applied Precision)

Support

BSD-licensed:

Export:

Officially Supported Versions:

Readers:

- DeltavisionReader (Source Code, *Supported Metadata Fields*)
- RCPNLReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- DeltaVision Opener plugin for ImageJ

We currently have:

- a DV specification document (v2.10 or newer, in HTML)
- numerous DV datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

- The Deltavision format is based on the Medical Research Council (MRC) file format.
- Commercial applications that support DeltaVision include:
  - Bitplane Imaris
  - SVI Huygens
  - Image-Pro Plus
- RCPNL is a variant of the DeltaVision format and requires a separate reader.

4.2.34 DICOM

Extensions: .dcm, .dicom

Developer: DICOM Standards Committee

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: DicomReader (Source Code, *Supported Metadata Fields*)

Writer: DicomWriter (Source Code)

Freely Available Software:

- List of converters and viewers
- List of libraries/toolkits
- Wikipedia's list of freeware health software

Sample Datasets:

- David Clunie's list of DICOM image samples
- Whole Slide Images from DICOM Working Group 26
- NCI Imaging Data Commons Portal, download instructions
- Converted Whole Slide Images
- DICOM sample image sets from OsiriX web site

We currently have:

- DICOM specification documents
- numerous DICOM datasets (see above)
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

- DICOM stands for "Digital Imaging and Communication in Medicine".
- Bio-Formats supports uncompressed, baseline JPEG, JPEG-2000 (lossy and lossless), and RLE lossless transfer syntaxes.
- Support for reading and writing DICOM whole slide images (DICOM WSI format) was implemented through collaboration with NCI Imaging Data Commons and has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, under Task Order No. HHSN26110071 under Contract No. HHSN2612015000031.

If you have a problematic DICOM file which you cannot send us for privacy reasons, please send us the exact error message and be aware that it may take several attempts to fix the problem blind.

See also

DICOM homepage

4.2.35 ECAT7

Extensions: .v

Developer: Siemens

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: Ecat7Reader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few ECAT7 files
- public sample images

We would like to have:

- an ECAT7 specification document
- more ECAT7 files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.36 EPS (Encapsulated PostScript)

Extensions: .eps, .epsi, .ps

Developer: Adobe

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: EPSReader (Source Code, *Supported Metadata Fields*)

Writer: EPSWriter (Source Code)

Freely Available Software:

- EPS Writer plugin for ImageJ

We currently have:

- a few EPS datasets
- the ability to produce new datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

- Bio-Formats can save individual planes as EPS.
- Certain types of compressed EPS files are not supported.

4.2.37 Evotec/PerkinElmer Opera Flex

Extensions: .flex, .mea, .res

Developer: Evotec Technologies, now PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: FlexReader (Source Code, *Supported Metadata Fields*)

We currently have:

- many Flex datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.38 FEI

Extensions: .img

Developer: FEI

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: FEIReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few FEI files

We would like to have:

- a specification document
- more FEI files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.39 FEI TIFF

Extensions: .tiff

Developer: FEI

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: FEITiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few FEI TIFF datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.40 FITS (Flexible Image Transport System)

Extensions: .fits

Developer: National Radio Astronomy Observatory

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: FitsReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a FITS specification document (NOST 100-2.0, from 1999 March 29, in HTML)
- several FITS datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

> **See also**
>
> MAST:FITS homepage FITS Support Office

4.2.41 Gatan Digital Micrograph

Extensions: .dm3, .dm4

Owner: Gatan

Support

BSD-licensed:

Export:

Officially Supported Versions: 3, 4

Reader: GatanReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- DM3 Reader plugin for ImageJ
- EMAN

We currently have:

- Gatan's ImageReader2003 code (from 2003, in C++)
- numerous DM3 datasets
- public sample images

We would like to have:

- a DM3 specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Applications that support .dm3 files include Datasqueeze.

Note that the Gatan Reader does not currently support stacks.

4.2.42 Gatan Digital Micrograph 2

Extensions: .dm2

Developer: Gatan

Support

BSD-licensed:

Export:

Officially Supported Versions: 2

Reader: GatanDM2Reader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read DM2 files (from ImageSXM)
- a few DM2 files

We would like to have:

- an official DM2 specification document
- more DM2 files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.43 GE MicroCT

Extensions: .vff

Developer: GE Healthcare

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MicroCTReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several public MicroCT datasets from the CIBC Dataset Archive

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.44 GIF (Graphics Interchange Format)

Extensions: .gif

Developer: CompuServe

Owner: Unisys

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: GIFReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Animated GIF Reader plugin for ImageJ
- GIF Stack Writer plugin for ImageJ

We currently have:

- a GIF specification document (Version 89a, from 1990, in HTML)
- numerous GIF datasets
- the ability to produce new datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.45 Hamamatsu Aquacosmos NAF

Extensions: .naf

Developer: Hamamatsu

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: NAFReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few NAF files

We would like to have:

- a specification document
- more NAF files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.46 Hamamatsu DCIMG

Extensions: .dcimg

Owner: Hamamatsu

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: DCIMGReader (Source Code, *Supported Metadata Fields*)

We currently have:

- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.47 Hamamatsu HIS

Extensions: .his

Owner: Hamamatsu

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: HISReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read HIS files (from ImageSXM)
- several HIS files

We would like to have:

- an HIS specification
- more HIS files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.48 Hamamatsu ndpi

Extensions: .ndpi, .ndpis

Developer: Hamamatsu

Support

BSD-licensed:

Export:

Officially Supported Versions:

Readers:

- NDPIReader (Source Code, *Supported Metadata Fields*)
- NDPISReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- NDP.view2
- OpenSlide

We currently have:

- many example datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.49 Hamamatsu VMS

Extensions: .vms

Developer: Hamamatsu

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: HamamatsuVMSReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- OpenSlide

We currently have:

- a few example datasets
- public sample images
- developer documentation from the OpenSlide project

We would like to have:

- an official specification document
- more example datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.50 Hitachi S-4800

Extensions: .txt, .tif, .bmp, .jpg

Developer: Hitachi

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: HitachiReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several Hitachi S-4800 datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.51 I2I

Extensions: .i2i

Developer: Biomedical Imaging Group, UMass Medical School

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: I2IReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several example datasets
- a specification document

- an ImageJ plugin that can read I2I data

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.52 ICS (Image Cytometry Standard)

Extensions: .ics, .ids

Developer: P. Dean et al.

Support

BSD-licensed:

Export:

Officially Supported Versions: 1.0, 2.0

Reader: ICSReader (Source Code, *Supported Metadata Fields*)

Writer: ICSWriter (Source Code)

Freely Available Software:

- Libics (ICS reference library)
- ICS Opener plugin for ImageJ
- IrfanView

We currently have:

- numerous ICS datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

- ICS version 1.0 datasets have two files - an .ics file that contains all of the metadata in plain-text format, and an .ids file that contains all of the pixel data.
- ICS version 2.0 datasets are a single .ics file that contains both pixels and metadata.

Commercial applications that can support ICS include:

- Bitplane Imaris

- SVI Huygens

4.2.53 Imacon

Extensions: .fff

Owner: Hasselblad

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ImaconReader (Source Code, *Supported Metadata Fields*)

We currently have:

- one Imacon file

We would like to have:

- more Imacon files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.54 ImagePro Sequence

Extensions: .seq

Owner: Media Cybernetics

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SEQReader (Source Code, *Supported Metadata Fields*)

We currently have:

- the Image-Pro Plus software
- a few SEQ datasets
- the ability to produce more datasets

We would like to have:

- an official SEQ specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.55 ImagePro Workspace

Extensions: .ipw

Owner: Media Cybernetics

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: IPWReader (Source Code, *Supported Metadata Fields*)

We currently have:

- the Image-Pro Plus software
- a few IPW datasets
- the ability to produce more datasets

We would like to have:

- an official IPW specification document
- more IPW datasets:
  - multiple datasets in one file
  - 2+ GB files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Bio-Formats uses a modified version of the Apache Jakarta POI library to read IPW files.

4.2.56 IMAGIC

Extensions: .hed, .img

Developer: Image Science

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ImagicReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- em2em

We currently have:

- one example dataset
- official file format documentation

We would like to have:

- more example datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

See also

IMAGIC specification

4.2.57 IMOD

Extensions: .mod

Developer: Boulder Laboratory for 3-Dimensional Electron Microscopy of Cells

Owner: Boulder Laboratory for 3-Dimensional Electron Microscopy of Cells

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: IMODReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- IMOD

We currently have:

- a few sample datasets
- official documentation

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.58 Improvision Openlab LIFF

Extensions: .liff

Developer: Improvision, now PerkinElmer

Owner: PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions: 2.0, 5.0

Reader: OpenlabReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an Openlab specification document (from 2000 February 8, in DOC)
- Improvision's XLIFFFileImporter code for reading Openlab LIFF v5 files (from 2006, in C++)
- several Openlab datasets

We would like to have:

- more Openlab datasets (preferably with 32-bit integer data)

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.59 Improvision Openlab Raw

Extensions: .raw

Developer: Improvision, now PerkinElmer

Owner: PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: OpenlabRawReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an Openlab Raw specification document (from 2004 November 09, in HTML)
- a few Openlab Raw datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.60 Improvision TIFF

Extensions: .tif

Developer: Improvision, now PerkinElmer

Owner: PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ImprovisionTiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an Improvision TIFF specification document
- a few Improvision TIFF datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.61 Imspector OBF

Extensions: .obf, .msr

Developer: Department of NanoBiophotonics, MPI-BPC

Owner: MPI-BPC

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: OBFReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few OBF datasets
- public OBF sample images
- a specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.62 InCell 1000/2000

Extensions: .xdce, .tif

Developer: GE Healthcare

Owner: Cytiva

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: InCellReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few InCell 1000 datasets
- public InCell 2000 sample images

We would like to have:

- an InCell 1000 specification document
- more InCell 1000 datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Additional options are available for reading or writing this format type, see *Additional reader and writer options* for information.

4.2.63 InCell 3000

Extensions: .frm

Developer: GE Healthcare

Owner: Cytiva

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: InCell3000Reader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few example datasets
- public sample images

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.64 INR

Extensions: .inr

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: INRReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several sample .inr datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.65 Inveon

Extensions: .hdr

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: InveonReader (Source Code, *Supported Metadata Fields*)

We currently have:

a few Inveon datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.66 Ionpath MIBI

Extensions: .tif, .tiff

Developer: *IonPath*

Owner: *IonPath*

Support

BSD-licensed:

Export:

Officially Supported Versions: 0.1

Reader: IonpathMIBITiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few sample datasets
- a specification document

We would like to have:

- more sample datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.67 IPLab

Extensions: .ipl

Developer: Scanalytics

Owner: was BD Biosystems, now BioVision Technologies

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: IPLabReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- IPLab Reader plugin for ImageJ

We currently have:

- an IPLab specification document (v3.6.5, from 2004 December 1, in PDF)
- several IPLab datasets

We would like to have:

- more IPLab datasets (preferably with 32-bit integer or floating point data)

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

Commercial applications that support IPLab include:

- Bitplane Imaris
- SVI Huygens

See also

IPLab software review

4.2.68 IVision

Extensions: .ipm

Owner: BioVision Technologies

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: IvisionReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few iVision-Mac datasets
- a specification document

We would like to have:

- more iVision-Mac datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

iVision-Mac was formerly called IPLab for Macintosh.

4.2.69 JDCE

Extensions: .jdce, .tif

Owner: Molecular Devices

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: JDCEReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few public JDCE plates: https://downloads.openmicroscopy.org/images/JDCE/

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.70 JEOL

Extensions: .dat, .img, .par

Owner: JEOL

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: JEOLReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that reads JEOL files (from ImageSXM)

- a few JEOL files

We would like to have:

- an official specification document
- more JEOL files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.71 JPEG

Extensions: .jpg

Developer: Independent JPEG Group

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: JPEGReader (Source Code, *Supported Metadata Fields*)

Writer: JPEGWriter (Source Code)

We currently have:

- a JPEG specification document (v1.04, from 1992 September 1, in PDF)
- numerous JPEG datasets
- the ability to produce more datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Bio-Formats can save individual planes as JPEG. Bio-Formats uses the Java Image I/O API to read and write JPEG files. JPEG stands for “Joint Photographic Experts Group”.

See also

JPEG homepage

4.2.72 JPEG 2000

Extensions: .jp2, .j2k, .jpf

Developer: Independent JPEG Group

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: JPEG2000Reader (Source Code, *Supported Metadata Fields*)

Writer: JPEG2000Writer (Source Code)

Freely Available Software:

- JJ2000 (JPEG 2000 library for Java)

We currently have:

- a JPEG 2000 specification document (free draft from 2000, no longer available online)
- a few .jp2 files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Bio-Formats uses the JAI Image I/O Tools library to read JP2 files. Conflicting versions of this no-longer-maintained library may cause errors, see *JAI ImageIO component* for details.

JPEG stands for “Joint Photographic Experts Group”.

4.2.73 JPK

Extensions: .jpk

Developer: JPK Instruments

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: JPKReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read JPK files (from ImageSXM)
- a few JPK files

We would like to have:

- an official specification document
- more JPK files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.74 JPX

Extensions: .jpx

Developer: JPEG Committee

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: JPXReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few .jpx files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.75 Keller Lab Block

Extensions: .klb

Developer: Keller Lab (Janelia Research Campus)

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: KLBReader (Source Code, *Supported Metadata Fields*)

We currently have:

- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.76 Khoros VIFF (Visualization Image File Format) Bitmap

Extensions: .xv

Developer: Khoral

Owner: AccuSoft

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: KhorosReader (Source Code, *Supported Metadata Fields*)

Sample Datasets:

- VIFF Images

We currently have:

- several VIFF datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.77 Kodak BIP

Extensions: .bip

Developer: Kodak/Carestream

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: KodakReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few .bip datasets

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.78 Lambert Instruments FLIM

Extensions: .fli

Developer: Lambert Instruments

Support

BSD-licensed:

Export:

Officially Supported Versions: 1.0, 2.0

Reader: LiFlimReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an LI-FLIM specification document
- several example LI-FLIM datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.79 LaVision Imspector

Extensions: .msr

Developer: LaVision BioTec

Support

BSD-licensed:

Export:

Officially Supported Versions: 4.0, 4.1

Reader: ImspectorReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few .msr files
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.80 Leica LCS LEI

Extensions: .lei, .tif

Developer: Leica Microsystems CMS GmbH

Owner: Leica Microsystems

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: LeicaReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Leica LCS Lite

We currently have:

- an LEI specification document (beta 2.000, from no later than 2004 February 17, in PDF)

- many LEI datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

LCS stands for "Leica Confocal Software". LEI presumably stands for "Leica Experimental Information".

Commercial applications that support LEI include:

- Bitplane Imaris
- SVI Huygens
- Image-Pro Plus

4.2.81 Leica LAS AF LIF (Leica Image File Format)

Extensions: .lif

Developer: Leica Microsystems CMS GmbH

Owner: Leica Microsystems

Support

BSD-licensed:

Export:

Officially Supported Versions: 1.0, 2.0

Reader: LIFReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Leica Application Suite X

We currently have:

- a LIF/XLLF/XLEF/LOF specification document (version 3.2, from no later than 2016 December 15, in PDF)
- a LIF specification document (version 2, from no later than 2007 July 26, in PDF)
- a LIF specification document (version 1, from no later than 2006 April 3, in PDF)
- numerous LIF datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

Additional options are available for reading or writing this format type, see *Additional reader and writer options* for information.

LAS stands for "Leica Application Suite". AF stands for "Advanced Fluorescence".

Commercial applications that support LIF include:

- Bitplane Imaris
- SVI Huygens
- Amira

Versions of Bio-Formats prior to 5.3.3 incorrectly calculated the physical pixel width and height. The physical image width and height were divided by the number of pixels, which was inconsistent with the official Leica LIF specification documents. Versions 5.3.3 and later correctly calculate physical pixel sizes by dividing the physical image size by the number of pixels minus one. To revert to the old method of physical pixel size calculation in 5.3.3 and later, set the `leicalif.old_physical_size` option to `true` as described in *Additional reader and writer options*.

4.2.82 Leica LOF

Extensions: .lof

Developer: Leica Microsystems CMS GmbH

Owner: Leica Microsystems

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: LOFReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Leica Application Suite X

We currently have:

- numerous LOF datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.83 Leica SCN

Extensions: .scn

Developer: Leica Microsystems

Support

BSD-licensed:

Export:

Officially Supported Versions: 2012-03-10

Reader: LeicaSCNReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- OpenSlide

We currently have:

- a few sample datasets
- public sample images

We would like to have:

- an official specification document
- sample datasets that cannot be opened

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.84 Leica XLEF

Extensions: .xlef

Developer: Leica Microsystems CMS GmbH

Owner: Leica Microsystems

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: XLEFReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Leica Application Suite X

We currently have:

- numerous XLEF datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.85 LEO

Extensions: .sxm, .tif, .tiff

Owner: ZEISS International

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: LEOReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read LEO files (from ImageSXM)
- a few LEO files
- public sample images

We would like to have:

- an official specification document
- more LEO files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.86 Li-Cor L2D

Extensions: .l2d, .tif, .scn

Owner: LiCor Biosciences

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: L2DReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few L2D datasets

We would like to have:

- an official specification document
- more L2D datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

L2D datasets cannot be imported into OME using server-side import. They can, however, be imported from ImageJ, or using the omeul utility.

4.2.87 LIM (Laboratory Imaging/Nikon)

Extensions: .lim

Owner: Laboratory Imaging

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: LIMReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several LIM files

- the ability to produce more LIM files

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Bio-Formats only supports uncompressed LIM files.

Commercial applications that support LIM include:

- NIS Elements

4.2.88 MetaMorph 7.5 TIFF

Extensions: .tiff

Owner: Molecular Devices

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MetamorphTiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few Metamorph 7.5 TIFF datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.89 MetaMorph Stack (STK)

Extensions: .stk, .nd

Owner: Molecular Devices

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MetamorphReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an STK specification document (from 2006 November 21, in DOC)
- an older STK specification document (from 2005 March 25, in DOC)
- an ND specification document (from 2002 January 24, in PDF)
- a large number of datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

Commercial applications that support STK include:

- Bitplane Imaris
- SVI Huygens
- DIMIN

See also

Metamorph imaging system overview

4.2.90 MetaXpress

Extensions: .htd, .tif, .tiff

Owner: Molecular Devices

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MetaxpressTiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several MetaXpress datasets
- public sample datasets

We would like to have:

- a MetaXpress specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.91 MIAS (Maia Scientific)

Extensions: .tif

Developer: Maia Scientific

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MIASReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several MIAS datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.92 Micro-Manager

Extensions: .tif, .txt, .xml

Developer: Vale Lab

Support

BSD-licensed:

Export:

Officially Supported Versions: Up to 1.4.22

Reader: MicromanagerReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Micro-Manager

We currently have:

- many Micro-manager datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

- Bio-Formats will recognize a `*metadata.txt` file as part of a Micro-Manager fileset if pointed at it and will load the fileset including the companion TIFF files.
- If pointed at a companion `.ome.tif` file, Bio-Formats will recognize an OME-TIFF format instead. This means it may load the fileset if there are multiple .ome.tif but it will not include `*metadata.txt` in this fileset and therefore the extended Micro-Manager metadata will be skipped.
- See Micro-Manager File Formats for more information.

4.2.93 Mikroscan TIFF

Extensions: .tif, .tiff

Owner: Mikroscan

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MikroscanTiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- some Mikroscan datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

4.2.94 MINC MRI

Extensions: .mnc

Developer: McGill University

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MINCReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- MINC

We currently have:

- a few MINC files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.95 Minolta MRW

Extensions: .mrw

Developer: Minolta

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MRWReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- dcraw

We currently have:

- several .mrw files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.96 MNG (Multiple-image Network Graphics)

Extensions: .mng

Developer: MNG Development Group

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MNGReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- libmng (MNG reference library)

Sample Datasets:

- MNG sample files

We currently have:

- the libmng-testsuites package (from 2003 March 05, in C)
- a large number of MNG datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

See also

MNG homepage MNG specification

4.2.97 Molecular Imaging

Extensions: .stp

Owner: Molecular Imaging Corp, San Diego CA (closed)

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MolecularImagingReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that reads Molecular Imaging files (from ImageSXM)
- a few Molecular Imaging files

We would like to have:

- an official specification document
- more Molecular Imaging files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.98 MRC (Medical Research Council)

Extensions: .mrc, .st, .ali, .map, .rec, .mrcs

Developers:

- MRC Laboratory of Molecular Biology
- CCP-EM

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: MRCReader (Source Code, *Supported Metadata Fields*)

Sample Datasets:

- golgi.mrc

We currently have:

- specification documents for MRC2000 and MRC2014
- an IMOD-specific MRC specification document (in TXT)

- public sample images
- a few MRC datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Commercial applications that support MRC include:

- Bitplane Imaris

Bio-Formats provides support for the base MRC2000/MRC2014 specifications. Limited support is provided for vendor-specific extended headers, as described in the MRC2014 specification. IMOD extended metadata is perhaps the best-supported variant.

See also

MRC on Wikipedia

4.2.99 NEF (Nikon Electronic Format)

Extensions: .nef, .tif

Developer: Nikon

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: NikonReader (Source Code, *Supported Metadata Fields*)

Sample Datasets:

- neffile1.zip
- weHave = * a NEF specification document (v0.1, from 2003, in PDF)
- – several NEF datasets

We currently have:

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

See also

NEF Conversion

4.2.100 NIfTI

Extensions: .img, .hdr, .nii, .nii.gz

Developer: National Institutes of Health

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: NiftiReader (Source Code, *Supported Metadata Fields*)

We currently have:

- NIfTI specification documents
- several NIfTI datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.101 Nikon Elements TIFF

Extensions: .tiff

Developer: Nikon

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: NikonElementsTiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few Nikon Elements TIFF files

We would like to have:

- more Nikon Elements TIFF files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.102 Nikon EZ-C1 TIFF

Extensions: .tiff

Developer: Nikon

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: NikonTiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few Nikon EZ-C1 TIFF files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.103 Nikon NIS-Elements ND2

Extensions: .nd2

Developer: Nikon USA

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ND2Reader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- NIS-Elements Viewer from Nikon

We currently have:

- many ND2 datasets
- public sample images

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Additional options are available for reading or writing this format type, see *Additional reader and writer options* for information.

There are two distinct versions of ND2: an old version, which uses JPEG-2000 compression, and a new version which is either uncompressed or Zip-compressed. We are not aware of the version number or release date for either format.

Bio-Formats uses the JAI Image I/O Tools library to read ND2 files compressed with JPEG-2000.

4.2.104 NRRD (Nearly Raw Raster Data)

Extensions: .nrrd, .nhdr, .raw, .txt

Developer: Teem developers

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: NRRDReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- nrrd (NRRD reference library)

Sample Datasets:

- Diffusion tensor MRI datasets

We currently have:

- an nrrd specification document (v1.9, from 2005 December 24, in HTML)
- a few nrrd datasets

- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.105 Olympus CellR/APL

Extensions: .apl, .mtb, .tnb, .tif, .obsep

Owner: Olympus

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: APLReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few CellR datasets

We would like to have:

- more Cellr datasets
- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.106 Olympus FluoView FV1000

Extensions: .oib, .oif

Owner: Olympus

Support

BSD-licensed:

Export:

Officially Supported Versions: 1.0, 2.0

Reader: FV1000Reader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- cellSens from Evident

We currently have:

- an OIF specification document (v2.0.0.0, from 2008, in PDF)
- an FV1000 specification document (v1.0.0.0, from 2004 June 22, in PDF)
- older FV1000 specification documents (draft, in DOC and XLS)
- public sample images
- many FV1000 datasets

We would like to have:

- more OIB datasets (especially 2+ GB files)
- more FV1000 version 2 datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

Bio-Formats uses a modified version of the Apache POI library to read OIB files. OIF stands for "Original Imaging Format". OIB stands for "Olympus Image Binary". OIF is a multi-file format that includes an .oif file and a directory of .tif, .roi, .pty, .lut, and .bmp files. OIB is a single file format.

Commercial applications that support this format include:

- Bitplane Imaris
- SVI Huygens

See also

Olympus FluoView Resource Center

4.2.107 Olympus FluoView TIFF

Extensions: .tif

Owner: Olympus

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: FluoviewReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- DIMIN

We currently have:

- a FluoView specification document (from 2002 November 14, in DOC)
- Olympus' FluoView Image File Reference Suite (from 2002 March 1, in DOC)
- several FluoView datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

Commercial applications that support this format include:

- Bitplane Imaris
- SVI Huygens

4.2.108 Olympus OIR

Extensions: .oir

Owner: Olympus

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: OIRReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Olympus Viewer Plugin for ImageJ

We currently have:

- several OIR datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Support for this format was added in partnership with OLYMPUS EUROPA SE & Co. KG

4.2.109 Olympus OMP2INFO

Extensions: .omp2info

Owner: Olympus

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: OlympusTileReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Olympus Viewer Plugin for ImageJ

We currently have:

- several OMP2INFO datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Support for this format was added in partnership with OLYMPUS EUROPA SE & Co. KG

4.2.110 Olympus ScanR

Extensions: .xml, .dat, .tif

Developer: Olympus

Owner: Olympus

Support

BSD-licensed:

Export:

Officially Supported Versions: Up to 2.5.1

Reader: ScanrReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several ScanR datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.111 Olympus SIS TIFF

Extensions: .tiff

Developer: Evident Scientific (formerly Olympus)

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SISReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few example SIS TIFF files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.112 OME-TIFF

Extensions: .ome.tiff, .ome.tif, .ome.tf2, .ome.tf8, .ome.btf

Developer: Open Microscopy Environment

Support

BSD-licensed:

Export:

Officially Supported Versions: 2003FC, 2007-06, 2008-02, 2008-09, 2009-09, 2010-04, 2010-06, 2011-06, 2012-06, 2013-06, 2015-01, 2016-06

Reader: OMETiffReader (Source Code, *Supported Metadata Fields*)

Writer: OMETiffWriter (Source Code)

We currently have:

- an OME-TIFF specification document
- many OME-TIFF datasets
- public sample images
- the ability to produce additional datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Additional options are available for reading or writing this format type, see *Additional reader and writer options* for information.

Bio-Formats can save image stacks as OME-TIFF.

Commercial applications that support OME-TIFF are listed on our commercial partners page

See also

OME-TIFF technical overview

4.2.113 OME-XML

Extensions: .ome, .ome.xml

Developer: Open Microscopy Environment

Support

BSD-licensed:

Export:

Officially Supported Versions: 2003FC, 2007-06, 2008-02, 2008-09, 2009-09, 2010-04, 2010-06, 2011-06, 2012-06, 2013-06, 2015-01, 2016-06

Reader: OMEXMLReader (Source Code, *Supported Metadata Fields*)

Writer: OMEXMLWriter (Source Code)

We currently have:

- OME-XML specification documents
- many OME-XML datasets
- public sample images
- the ability to produce more datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Bio-Formats uses the OME-XML Java library to read OME-XML files.

Commercial applications that support OME-XML include:

- Bitplane Imaris
- SVI Huygens

4.2.114 OMERO Pyramid

Developer: Open Microscopy Environment

Support

BSD-licensed:

Export:

Officially Supported Versions: 1.0.0

Reader: TiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a *specification document*
- several OMERO Pyramid datasets
- the ability to produce additional datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.115 Oxford Instruments

Extensions: .top

Owner: Oxford Instruments

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: OxfordInstrumentsReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read Oxford Instruments files (from ImageSXM)
- a few Oxford Instruments files

We would like to have:

- an official specification document
- more Oxford Instruments files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.116 PCORAW

Extensions: .pcoraw, .rec

Developer: PCO

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PCORAWReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few example datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.117 PCX (PC Paintbrush)

Extensions: .pcx

Developer: ZSoft Corporation

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PCXReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several .pcx files
- the ability to generate additional .pcx files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Commercial applications that support PCX include Zeiss LSM Image Browser.

4.2.118 Perkin Elmer Densitometer

Extensions: .pds

Developer: Perkin Elmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PDSReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few PDS datasets

We would like to have:

- an official specification document
- more PDS datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.119 PerkinElmer Columbus

Extensions: .xml, .csv, .tif

Owner: PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ColumbusReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few example datasets
- public sample images

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.120 PerkinElmer Nuance

Extensions: .im3

Developer: PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: IM3Reader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few sample datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.121 PerkinElmer Operetta

Extensions: .tiff, .xml

Developer: PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: OperettaReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few sample datasets
- public sample images

We would like to have:

- an official specification document
- more sample datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.122 PerkinElmer UltraVIEW

Extensions: .tif, .2, .3, .4, etc.

Owner: PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PerkinElmerReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several UltraVIEW datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Other associated extensions include: .tim, .zpo, .csv, .htm, .cfg, .ano, .rec

Commercial applications that support this format include:

- Bitplane Imaris
- Image-Pro Plus

See also

PerkinElmer UltraVIEW system overview (pdf)

4.2.123 Portable Any Map

Extensions: .pbm, .pgm, .ppm

Developer: Netpbm developers

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PGMReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Netpbm graphics filter

We currently have:

- a PGM specification document (from 2003 October 3, in HTML)
- a few PBM, PPM and PGM files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.124 Adobe Photoshop PSD

Extensions: .psd

Developer: Adobe

Support

BSD-licensed:

Export:

Officially Supported Versions: 1.0

Reader: PSDReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a PSD specification document (v3.0.4, 16 July 1995)
- a few PSD files

We would like to have:

- more PSD files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.125 Photoshop TIFF

Extensions: .tif, .tiff

Developer: Adobe

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PhotoshopTiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a Photoshop TIFF specification document
- a few Photoshop TIFF files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.126 PicoQuant Bin

Extensions: .bin

Developer: PicoQuant

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PQBinReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- SymphoTime64

We currently have:

- a few example datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.127 PICT (Macintosh Picture)

Extensions: .pict

Developer: Apple Computer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PictReader (Source Code, *Supported Metadata Fields*)

We currently have:

- many PICT datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.128 PNG (Portable Network Graphics)

Extensions: .png

Developer: PNG Development Group

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: APNGReader (Source Code, *Supported Metadata Fields*)

Writer: APNGWriter (Source Code)

Freely Available Software:

- PNG Writer plugin for ImageJ

We currently have:

- a PNG specification document (W3C/ISO/IEC version, from 2003 November 10, in HTML)
- several PNG datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Bio-Formats uses the Java Image I/O API to read and write PNG files.

See also

PNG technical overview

4.2.129 Prairie Technologies TIFF

Extensions: .tif, .xml, .cfg

Developer: Prairie Technologies

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PrairieReader (Source Code, *Supported Metadata Fields*)

We currently have:

- many Prairie datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.130 Princeton Instruments SPE

Extensions: .spe

Developer: Princeton Instruments

Support

BSD-licensed:

Export:

Officially Supported Versions: 3.0

Reader: SPEReader (Source Code, *Supported Metadata Fields*)

We currently have:

- An official specification document
- two SPE files

We would like to have:

- more SPE files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.131 Quesant

Extensions: .afm

Developer: Quesant Instrument Corporation

Owner: KLA-Tencor Corporation

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: QuesantReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read Quesant files (from ImageSXM)
- several Quesant files

We would like to have:

- an official specification document
- more Quesant files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.132 QuickTime Movie

Extensions: .mov

Owner: Apple Computer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: QTReader (Source Code, *Supported Metadata Fields*)

Writer: QTWriter (Source Code)

Freely Available Software:

- QuickTime Player

We currently have:

- a QuickTime specification document
- several QuickTime datasets
- the ability to produce more datasets

We would like to have:

- more QuickTime datasets, including:
  - files compressed with a common, unsupported codec
  - files with audio tracks and/or multiple video tracks

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Bio-Formats can save image stacks as QuickTime movies. The following table shows supported codecs:

Codec	Description	Native
raw	Full Frames (Uncompressed)	read & write
iraw	Intel YUV Uncompressed	read only
rle	Animation (run length encoded RGB)	read only
jpeg	Still Image JPEG DIB	read only
rpza	Apple Video 16 bit “road pizza”	read only (partial)
mjpb	Motion JPEG codec	read only

See also

QuickTime software overview

4.2.133 RHK

Extensions: .sm2, .sm3

Owner: RHK Technologies

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: RHKReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read RHK files (from ImageSXM)
- a few RHK files

We would like to have:

- an official specification document
- more RHK files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.134 SBIG

Owner: Diffraction Limited (formerly Santa Barbara Instrument Group)

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SBIGReader (Source Code, *Supported Metadata Fields*)

We currently have:

- an official SBIG specification document
- a few SBIG files

We would like to have:

- more SBIG files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.135 Seiko

Extensions: .xqd, .xqf

Owner: Seiko

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SeikoReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read Seiko files (from ImageSXM)
- a few Seiko files

We would like to have:

- an official specification document
- more Seiko files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.136 SimplePCI & HCImage

Extensions: .cxd

Developer: Compix

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: PCIReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several SimplePCI files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Bio-Formats uses a modified version of the Apache POI library to read CXD files.

See also

SimplePCI software overview

4.2.137 SimplePCI & HCImage TIFF

Extensions: .tiff

Developer: Hamamatsu

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SimplePCITiffReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few SimplePCI TIFF datasets

We would like to have:

- more SimplePCI TIFF datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.138 SM Camera

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SMCameraReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read SM-Camera files (from ImageSXM)

- a few SM-Camera files

We would like to have:

- an official specification document
- more SM-Camera files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.139 SPIDER

Extensions: .spi, .stk

Developer: Wadsworth Center

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: SpiderReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- SPIDER

We currently have:

- a few example datasets
- official file format documentation

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.140 Targa

Extensions: .tga

Developer: Truevision

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: TargaReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a Targa specification document
- a few Targa files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.141 Tecan Spark Cyto Workspace

Extensions: .db, .tif

Owner: Tecan Trading

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: TecanReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few Tecan Spark Cyto workspaces

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Fiji users will need to enable the “Tissue Analyzer” update site and install “sqlite-jdbc.jar”

4.2.142 Text

Extensions: .txt

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: TextReader (Source Code, *Supported Metadata Fields*)

We currently have:

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Reads tabular pixel data produced by a variety of software.

4.2.143 TIFF (Tagged Image File Format)

Extensions: .tiff, .tif, .tf2, .tf8, .btf

Developer: Aldus and Microsoft

Owner: Adobe

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: TiffReader (Source Code, *Supported Metadata Fields*)

Writer: TiffWriter (Source Code)

Sample Datasets:

- Big TIFF

We currently have:

- TIFF specification documents from Adobe
- a TIFF specification document
- public sample images
- many TIFF datasets
- a few BigTIFF datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Bio-Formats can also read BigTIFF files (TIFF files larger than 4 GB). Bio-Formats can save image stacks as TIFF or BigTIFF.

TIFF files written by ImageJ are also supported, including ImageJ TIFFs larger than 4GB. ImageJ TIFFs are detected based upon the text in the first IFD's "ImageDescription" tag; this tag's value is then used to determine Z, C, and T sizes as well as physical sizes and timestamps. For ImageJ TIFFs larger than 4GB, a single IFD is expected (instead of one IFD per image plane). The "ImageDescription" is used to determine the number of images, the pixel data for which are expected to be stored contiguously at the offset indicated in the sole IFD. This differs from standard TIFF and BigTIFF; if the "ImageDescription" tag is missing or invalid, only the first image will be read.

See also

TIFF technical overview BigTIFF technical overview ImageJ TIFF overview Source code for ImageJ's native TIFF reader

4.2.144 TillPhotonics TillVision

Extensions: .vws

Developer: TILL Photonics

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: TillVisionReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several TillVision datasets

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.145 TissueGnostics TFCYTO

Extensions: .tfcyto, .aqproj

Owner: TissueGnostics

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: TissueFAXSReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a specification document
- several TFCYTO datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.146 Topometrix

Extensions: .tfr, .ffr, .zfr, .zfp, .2fl

Owner: TopoMetrix (now Veeco)

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: TopometrixReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that reads Topometrix files (from ImageSXM)
- a few Topometrix files

We would like to have:

- an official specification document
- more Topometrix files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.147 Trestle

Extensions: .tif, .sld, .jpg

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: TrestleReader (Source Code, *Supported Metadata Fields*)

Sample Datasets:

- OpenSlide

We currently have:

- a few example datasets
- public sample images
- developer documentation from the OpenSlide project

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.148 UBM

Extensions: .pr3

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: UBMReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read UBM files (from ImageSXM)
- one UBM file

We would like to have:

- an official specification document
- more UBM files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.149 Unisoku

Extensions: .dat, .hdr

Owner: Unisoku

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: UnisokuReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read Unisoku files (from ImageSXM)
- a few Unisoku files

We would like to have:

- an official specification document
- more Unisoku files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.150 Varian FDF

Extensions: .fdf

Developer: Varian, Inc.

Owner: Agilent Technologies

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: VarianFDFReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few Varian FDF datasets

We would like to have:

- an official specification document
- more Varian FDF datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.151 Vectra QPTIFF

Extensions: .tif, .qptiff

Owner: Akoya Biosciences, formerly owned by PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: VectraReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a specification document
- public sample images
- several datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Support for this format was added in partnership with PerkinElmer and updated in partnership with Akoya Biosciences

4.2.152 Veeco AFM

Extensions: .hdf

Developer: Veeco

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: VeecoReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few sample datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.153 Ventana BIF

Extensions: .bif

Owner: Roche Digital Diagnostics

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: VentanaReader (Source Code, *Supported Metadata Fields*)

We currently have:

- some Ventana datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.154 VG SAM

Extensions: .dti

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: VGSAMReader (Source Code, *Supported Metadata Fields*)

We currently have:

- a few VG-SAM files

We would like to have:

- an official specification document
- more VG-SAM files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.155 VisiTech XYS

Extensions: .xys, .html

Developer: VisiTech International

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: VisitechReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several VisiTech datasets

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.156 Volocity

Extensions: .mvd2

Developer: PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: VolocityReader (Source Code, *Supported Metadata Fields*)

Sample Datasets:

- PerkinElmer Downloads

We currently have:

- many example Volocity datasets

We would like to have:

- an official specification document
- any Volocity datasets that do not open correctly

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

.mvd2 files are Metakit database files.

4.2.157 Volocity Library Clipping

Extensions: .acff

Developer: PerkinElmer

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: VolocityClippingReader (Source Code, *Supported Metadata Fields*)

We currently have:

- several Volocity library clipping datasets

We would like to have:

- any datasets that do not open correctly
- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

RGB .acff files are not yet supported. See #6413.

4.2.158 WA-TOP

Extensions: .wat

Developer: WA Technology

Owner: Oxford Instruments

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: WATOPReader (Source Code, *Supported Metadata Fields*)

We currently have:

- Pascal code that can read WA-TOP files (from ImageSXM)
- a few WA-TOP files

We would like to have:

- an official specification document

- more WA-TOP files

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.159 Windows Bitmap

Extensions: .bmp

Developer: Microsoft and IBM

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: BMPReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- BMP Writer plugin for ImageJ

We currently have:

- many BMP datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Compressed BMP files are currently not supported.

See also

Technical Overview

4.2.160 Zeiss Axio CSM

Extensions: .lms

Developer: ZEISS International

Owner: ZEISS International

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ZeissLMSReader (Source Code, *Supported Metadata Fields*)

We currently have:

- one example dataset

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

This should not be confused with the more common Zeiss LSM format, which has a similar extension. As far as we know, the Axio CSM 700 system is the only one which saves files in the .lms format.

4.2.161 Zeiss AxioVision TIFF

Extensions: .xml, .tif

Developer: ZEISS International

Owner: ZEISS International

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ZeissTIFFReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Zeiss ZEN Lite

We currently have:

- many example datasets

We would like to have:

- an official specification document

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.2.162 Zeiss AxioVision ZVI (Zeiss Vision Image)

Extensions: .zvi

Developer: Carl Zeiss Microscopy GmbH

Owner: ZEISS International

Support

BSD-licensed:

Export:

Officially Supported Versions: 1.0, 2.0

Reader: ZeissZVIReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Zeiss Zen

We currently have:

- a ZVI specification document (v2.0.5, from 2010 August, in PDF)
- an older ZVI specification document (v2.0.2, from 2006 August 23, in PDF)
- an older ZVI specification document (v2.0.1, from 2005 April 21, in PDF)
- an older ZVI specification document (v1.0.26.01.01, from 2001 January 29, in DOC)
- Zeiss' ZvImageReader code (v1.0, from 2001 January 25, in C++)
- many ZVI datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

Bio-Formats uses a modified version of the Apache POI library to read ZVI files. ImageJ/FIJI will use the ZVI reader plugin in preference to Bio-Formats if both are installed. If you have a problem which is solved by opening the file using the Bio-Formats Importer plugin, you can just remove the ZVI_Reader.class from the plugins folder.

Commercial applications that support ZVI include Bitplane Imaris.

As of May 2021, the proprietary ZEISS AxioVision software is classed as End of Support. Zeiss ZEN is the successor programme to AxioVision.

4.2.163 Zeiss CZI

Extensions: .czi

Developer: ZEISS International

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ZeissCZIReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Zeiss ZEN
- libCZI

We currently have:

- many example datasets
- public sample images

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

Additional options are available for reading or writing this format type, see *Additional reader and writer options* for information.

JPEG-XR compressed CZI files are supported on the following 64-bit platforms:

- Windows 7 and above with Visual Studio 2015 C++ Redistributable
- CentOS 6 and above, Ubuntu 12.04 and above
- OS X 10.10 and above

4.2.164 Zeiss LSM (Laser Scanning Microscope) 510/710

Extensions: .lsm, .mdb

Owner: ZEISS International

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ZeissLSMReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- Zeiss LSM Image Browser
- LSM Toolbox plugin for ImageJ
- LSM Reader plugin for ImageJ
- DIMIN

We currently have:

- LSM specification v3.2, from 2003 March 12, in PDF
- LSM specification v5.5, from 2009 November 23, in PDF
- LSM specification v6.0, from 2010 September 28, in PDF
- many LSM datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

Additional Information

Please note that while we have specification documents for this format, we are not able to distribute them to third parties.

Bio-Formats uses the OME MDB Tools Java port. This is a fork of the mdbtools-java project.

Commercial applications that support this format include:

- SVI Huygens
- Bitplane Imaris
- Amira
- Image-Pro Plus

4.2.165 Zeiss XRM

Extensions: .txm, .txrm

Developer: ZEISS X-Ray Microscopy division

Owner: ZEISS International

Support

BSD-licensed:

Export:

Officially Supported Versions:

Reader: ZeissXRMReader (Source Code, *Supported Metadata Fields*)

Freely Available Software:

- XRM Data Explorer

We currently have:

- a few sample datasets

Ratings

Pixels:

Metadata:

Openness:

Presence:

Utility:

4.3 Summary of supported metadata fields

4.3.1 Format readers

Reader	Supported	Unsupported	Partial	Unknown/Missing
AFIReader	31	0	0	445
AIMReader	22	0	0	454
APLReader	21	0	0	455
APNGReader	19	0	0	457
ARFReader	19	0	0	457
AVIReader	19	0	0	457
AliconaReader	33	0	0	443
AmiraReader	22	0	0	454
AnalyzeReader	24	0	0	452
BDReader	59	0	0	417
BDVReader	37	0	0	439
BIFormatReader	19	0	0	457
BMPReader	21	0	0	455
BaseTiffReader	28	0	0	448
BaseZeissReader	84	0	0	392

continues on next page

Table 3 – continued from previous page

Reader	Supported	Unsupported	Partial	Unknown/Missing
BioRadGelReader	21	0	0	455
BioRadReader	40	0	0	436
BioRadSCNReader	29	0	0	447
BrukerReader	23	0	0	453
BurleighReader	22	0	0	454
CV7000Reader	57	0	0	419
CanonRawReader	19	0	0	457
CellH5Reader	41	0	0	435
CellSensReader	49	0	0	427
CellVoyagerReader	39	0	0	437
CellWorxReader	50	0	0	426
CellomicsReader	36	0	0	440
ColumbusReader	43	0	0	433
DCIMGReader	19	0	0	457
DNGReader	19	0	0	457
DeltavisionReader	51	0	0	425
DicomReader	27	0	0	449
EPSReader	19	0	0	457
Ecat7Reader	23	0	0	453
FEIReader	19	0	0	457
FEITiffReader	39	0	0	437
FV1000Reader	113	0	0	363
FakeReader	101	0	0	375
FilePatternReader	20	0	0	456
FitsReader	19	0	0	457
FlexReader	71	0	0	405
FlowSightReader	20	0	0	456
FluoviewReader	49	0	0	427
FujiReader	23	0	0	453
GIFReader	19	0	0	457
GatanDM2Reader	30	0	0	446
GatanReader	69	0	0	407
GelReader	21	0	0	455
HISReader	27	0	0	449
HRDGDFReader	21	0	0	455
HamamatsuVMSReader	26	0	0	450
HitachiReader	31	0	0	445
I2IReader	19	0	0	457
ICSReader	72	0	0	404
IM3Reader	19	0	0	457
IMODReader	44	0	0	432
INRReader	22	0	0	454
IPLabReader	31	0	0	445
IPWReader	20	0	0	456
ImaconReader	23	0	0	453
ImageIOReader	19	0	0	457
ImagicReader	22	0	0	454
ImarisHDFReader	24	0	0	452
ImarisReader	32	0	0	444

continues on next page

Table 3 – continued from previous page

Reader	Supported	Unsupported	Partial	Unknown/Missing
ImarisTiffReader	23	0	0	453
ImprovisionTiffReader	26	0	0	450
ImspectorReader	19	0	0	457
InCell3000Reader	19	0	0	457
InCellReader	69	0	0	407
InveonReader	30	0	0	446
IonpathMIBITiffReader	22	0	0	454
IvisionReader	34	0	0	442
JDCEReader	39	0	0	437
JEOLReader	19	0	0	457
JPEG2000Reader	19	0	0	457
JPEGReader	19	0	0	457
JPKReader	19	0	0	457
JPXReader	19	0	0	457
KLBReader	22	0	0	454
KhorosReader	19	0	0	457
KodakReader	26	0	0	450
L2DReader	29	0	0	447
LEOReader	27	0	0	449
LIFReader	85	0	0	391
LIMReader	19	0	0	457
LOFReader	19	0	0	457
LeicaReader	56	0	0	420
LeicaSCNReader	33	0	0	443
LiFlimReader	25	0	0	451
MIASReader	67	0	0	409
MINCReader	26	0	0	450
MNGReader	19	0	0	457
MRCReader	22	0	0	454
MRWReader	19	0	0	457
MetamorphReader	59	0	0	417
MetamorphTiffReader	43	0	0	433
MetaxpressTiffReader	19	0	0	457
MicroCTReader	24	0	0	452
MicromanagerReader	42	0	0	434
MikroscanTiffReader	19	0	0	457
MinimalTiffReader	19	0	0	457
MolecularImagingReader	21	0	0	455
NAFReader	19	0	0	457
ND2Reader	52	0	0	424
NDPIReader	33	0	0	443
NDPISReader	22	0	0	454
NRRDReader	22	0	0	454
NiftiReader	24	0	0	452
NikonElementsTiffReader	50	0	0	426
NikonReader	19	0	0	457
NikonTiffReader	47	0	0	429
OBFReader	19	0	0	457
OIRReader	48	0	0	428

continues on next page

Table 3 – continued from previous page

Reader	Supported	Unsupported	Partial	Unknown/Missing
OMETiffReader	19	0	0	457
OMEXMLReader	19	0	0	457
OlympusTileReader	19	0	0	457
OpenlabRawReader	19	0	0	457
OpenlabReader	32	0	0	444
OperettaReader	60	0	0	416
OxfordInstrumentsReader	22	0	0	454
PCIReader	29	0	0	447
PCORAWReader	26	0	0	450
PCXReader	19	0	0	457
PDSReader	23	0	0	453
PGMReader	19	0	0	457
PQBinReader	21	0	0	455
PSDReader	19	0	0	457
PerkinElmerReader	30	0	0	446
PhotoshopTiffReader	19	0	0	457
PictReader	19	0	0	457
PovrayReader	19	0	0	457
PrairieReader	46	0	0	430
PyramidTiffReader	19	0	0	457
QTReader	19	0	0	457
QuesantReader	22	0	0	454
RCPNLReader	26	0	0	450
RHKReader	22	0	0	454
SBIGReader	22	0	0	454
SDTReader	19	0	0	457
SEQReader	20	0	0	456
SIFReader	20	0	0	456
SISReader	33	0	0	443
SMCameraReader	19	0	0	457
SPCReader	19	0	0	457
SPEReader	30	0	0	446
SVSReader	33	0	0	443
ScanrReader	43	0	0	433
SeikoReader	22	0	0	454
SimplePCITiffReader	33	0	0	443
SlideBook7Reader	64	0	0	412
SlidebookReader	34	0	0	442
SlidebookTiffReader	30	0	0	446
SpiderReader	21	0	0	455
TCSReader	24	0	0	452
TargaReader	20	0	0	456
TecanReader	36	0	0	440
TextReader	19	0	0	457
TiffDelegateReader	19	0	0	457
TiffJAIReader	19	0	0	457
TiffReader	22	0	0	454
TileJPEGReader	19	0	0	457
TillVisionReader	22	0	0	454

continues on next page

Table 3 – continued from previous page

Reader	Supported	Unsupported	Partial	Unknown/Missing
TissueFAXSReader	33	0	0	443
TopometrixReader	22	0	0	454
TrestleReader	27	0	0	449
UBMReader	19	0	0	457
UnisokuReader	22	0	0	454
VGSAMReader	19	0	0	457
VarianFDFReader	25	0	0	451
VectraReader	43	0	0	433
VeecoReader	19	0	0	457
VentanaReader	28	0	0	448
VisitechReader	19	0	0	457
VolocityClippingReader	19	0	0	457
VolocityReader	38	0	0	438
WATOPReader	22	0	0	454
XLEFReader	19	0	0	457
ZeissCZIReader	181	0	0	295
ZeissLMSReader	23	0	0	453
ZeissLSMReader	101	0	0	375
ZeissTIFFReader	19	0	0	457
ZeissXRMReader	27	0	0	449
ZeissZVIReader	19	0	0	457
ZipReader	19	0	0	457

4.3.2 Metadata fields

Field	Supported	Unsupported	Partial	Unknown/Missing
Arc - ID	1	0	0	183
Arc - LotNumber	1	0	0	183
Arc - Manufacturer	1	0	0	183
Arc - Model	1	0	0	183
Arc - Power	1	0	0	183
Arc - SerialNumber	1	0	0	183
Arc - Type	0	0	0	184
BooleanAnnotation - AnnotationRef	0	0	0	184
BooleanAnnotation - Description	0	0	0	184
BooleanAnnotation - ID	1	0	0	183
BooleanAnnotation - Namespace	1	0	0	183
BooleanAnnotation - Value	1	0	0	183
Channel - AcquisitionMode	6	0	0	178
Channel - AnnotationRef	0	0	0	184
Channel - Color	21	0	0	163
Channel - ContrastMethod	1	0	0	183
Channel - EmissionWavelength	28	0	0	156
Channel - ExcitationWavelength	24	0	0	160
Channel - FilterSetRef	2	0	0	182
Channel - Fluor	3	0	0	181
Channel - ID	184	0	0	0

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsupported	Partial	Unknown/Missing
Channel - IlluminationType	3	0	0	181
Channel - LightSourceSettingsAttenuation	2	0	0	182
Channel - LightSourceSettingsID	9	0	0	175
Channel - LightSourceSettingsWavelength	2	0	0	182
Channel - NDFilter	2	0	0	182
Channel - Name	51	0	0	133
Channel - PinholeSize	11	0	0	173
Channel - PockelCellSetting	0	0	0	184
Channel - SamplesPerPixel	184	0	0	0
CommentAnnotation - AnnotationRef	0	0	0	184
CommentAnnotation - Description	0	0	0	184
CommentAnnotation - ID	1	0	0	183
CommentAnnotation - Namespace	1	0	0	183
CommentAnnotation - Value	1	0	0	183
Dataset - AnnotationRef	0	0	0	184
Dataset - Description	0	0	0	184
Dataset - ExperimenterGroupRef	0	0	0	184
Dataset - ExperimenterRef	0	0	0	184
Dataset - ID	0	0	0	184
Dataset - ImageRef	0	0	0	184
Dataset - Name	0	0	0	184
Detector - AmplificationGain	2	0	0	182
Detector - AnnotationRef	0	0	0	184
Detector - Gain	7	0	0	177
Detector - ID	37	0	0	147
Detector - LotNumber	1	0	0	183
Detector - Manufacturer	5	0	0	179
Detector - Model	15	0	0	169
Detector - Offset	7	0	0	177
Detector - SerialNumber	4	0	0	180
Detector - Type	28	0	0	156
Detector - Voltage	3	0	0	181
Detector - Zoom	4	0	0	180
DetectorSettings - Binning	19	0	0	165
DetectorSettings - Gain	21	0	0	163
DetectorSettings - ID	36	0	0	148
DetectorSettings - Offset	9	0	0	175
DetectorSettings - ReadOutRate	5	0	0	179
DetectorSettings - Voltage	6	0	0	178
Dichroic - AnnotationRef	0	0	0	184
Dichroic - ID	6	0	0	178
Dichroic - LotNumber	1	0	0	183
Dichroic - Manufacturer	1	0	0	183
Dichroic - Model	6	0	0	178
Dichroic - SerialNumber	1	0	0	183
DoubleAnnotation - AnnotationRef	0	0	0	184
DoubleAnnotation - Description	0	0	0	184
DoubleAnnotation - ID	1	0	0	183
DoubleAnnotation - Namespace	1	0	0	183

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsupported	Partial	Unknown/Missing
DoubleAnnotation - Value	1	0	0	183
Ellipse - FillColor	0	0	0	184
Ellipse - FillRule	0	0	0	184
Ellipse - FontFamily	0	0	0	184
Ellipse - FontSize	3	0	0	181
Ellipse - FontStyle	0	0	0	184
Ellipse - ID	8	0	0	176
Ellipse - Locked	0	0	0	184
Ellipse - RadiusX	8	0	0	176
Ellipse - RadiusY	8	0	0	176
Ellipse - StrokeColor	1	0	0	183
Ellipse - StrokeDashArray	0	0	0	184
Ellipse - StrokeWidth	2	0	0	182
Ellipse - Text	4	0	0	180
Ellipse - TheC	0	0	0	184
Ellipse - TheT	2	0	0	182
Ellipse - TheZ	2	0	0	182
Ellipse - Transform	2	0	0	182
Ellipse - X	8	0	0	176
Ellipse - Y	8	0	0	176
Experiment - AnnotationRef	0	0	0	184
Experiment - Description	1	0	0	183
Experiment - ExperimenterRef	0	0	0	184
Experiment - ID	5	0	0	179
Experiment - Type	5	0	0	179
Experimenter - AnnotationRef	0	0	0	184
Experimenter - Email	2	0	0	182
Experimenter - FirstName	5	0	0	179
Experimenter - ID	12	0	0	172
Experimenter - Institution	4	0	0	180
Experimenter - LastName	9	0	0	175
Experimenter - MiddleName	1	0	0	183
Experimenter - UserName	4	0	0	180
ExperimenterGroup - AnnotationRef	0	0	0	184
ExperimenterGroup - Description	0	0	0	184
ExperimenterGroup - ExperimenterRef	0	0	0	184
ExperimenterGroup - ID	0	0	0	184
ExperimenterGroup - Leader	0	0	0	184
ExperimenterGroup - Name	0	0	0	184
Filament - ID	1	0	0	183
Filament - LotNumber	1	0	0	183
Filament - Manufacturer	1	0	0	183
Filament - Model	1	0	0	183
Filament - Power	1	0	0	183
Filament - SerialNumber	1	0	0	183
Filament - Type	0	0	0	184
FileAnnotation - AnnotationRef	0	0	0	184
FileAnnotation - Description	0	0	0	184
FileAnnotation - ID	0	0	0	184

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsup-ported	Partial	Un-known/Missing
FileAnnotation - Namespace	0	0	0	184
Filter - AnnotationRef	0	0	0	184
Filter - FilterWheel	2	0	0	182
Filter - ID	8	0	0	176
Filter - LotNumber	1	0	0	183
Filter - Manufacturer	1	0	0	183
Filter - Model	8	0	0	176
Filter - SerialNumber	1	0	0	183
Filter - Type	2	0	0	182
FilterSet - DichroicRef	2	0	0	182
FilterSet - EmissionFilterRef	2	0	0	182
FilterSet - ExcitationFilterRef	2	0	0	182
FilterSet - ID	2	0	0	182
FilterSet - LotNumber	1	0	0	183
FilterSet - Manufacturer	1	0	0	183
FilterSet - Model	2	0	0	182
FilterSet - SerialNumber	1	0	0	183
Folder - AnnotationRef	0	0	0	184
Folder - Description	0	0	0	184
Folder - FolderRef	0	0	0	184
Folder - ID	0	0	0	184
Folder - ImageRef	0	0	0	184
Folder - Name	0	0	0	184
Folder - ROIRef	0	0	0	184
Image - AcquisitionDate	184	0	0	0
Image - AnnotationRef	1	0	0	183
Image - Description	48	0	0	136
Image - ExperimentRef	2	0	0	182
Image - ExperimenterGroupRef	0	0	0	184
Image - ExperimenterRef	7	0	0	177
Image - ID	184	0	0	0
Image - InstrumentRef	53	0	0	131
Image - MicrobeamManipulationRef	0	0	0	184
Image - Name	184	0	0	0
Image - ROIRef	17	0	0	167
ImagingEnvironment - AirPressure	1	0	0	183
ImagingEnvironment - CO2Percent	1	0	0	183
ImagingEnvironment - Humidity	1	0	0	183
ImagingEnvironment - Temperature	10	0	0	174
Instrument - AnnotationRef	0	0	0	184
Instrument - ID	59	0	0	125
Label - FillColor	0	0	0	184
Label - FillRule	0	0	0	184
Label - FontFamily	0	0	0	184
Label - FontSize	3	0	0	181
Label - FontStyle	0	0	0	184
Label - ID	6	0	0	178
Label - Locked	0	0	0	184
Label - StrokeColor	1	0	0	183

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsupported	Partial	Unknown/Missing
Label - StrokeDashArray	0	0	0	184
Label - StrokeWidth	2	0	0	182
Label - Text	6	0	0	178
Label - TheC	0	0	0	184
Label - TheT	0	0	0	184
Label - TheZ	0	0	0	184
Label - Transform	0	0	0	184
Label - X	6	0	0	178
Label - Y	6	0	0	178
Laser - FrequencyMultiplication	0	0	0	184
Laser - ID	12	0	0	172
Laser - LaserMedium	8	0	0	176
Laser - LotNumber	1	0	0	183
Laser - Manufacturer	2	0	0	182
Laser - Model	5	0	0	179
Laser - PockelCell	0	0	0	184
Laser - Power	4	0	0	180
Laser - Pulse	0	0	0	184
Laser - Pump	0	0	0	184
Laser - RepetitionRate	1	0	0	183
Laser - SerialNumber	1	0	0	183
Laser - Tuneable	0	0	0	184
Laser - Type	8	0	0	176
Laser - Wavelength	9	0	0	175
LightEmittingDiode - ID	1	0	0	183
LightEmittingDiode - LotNumber	1	0	0	183
LightEmittingDiode - Manufacturer	1	0	0	183
LightEmittingDiode - Model	1	0	0	183
LightEmittingDiode - Power	1	0	0	183
LightEmittingDiode - SerialNumber	1	0	0	183
LightPath - AnnotationRef	0	0	0	184
LightPath - DichroicRef	4	0	0	180
LightPath - EmissionFilterRef	6	0	0	178
LightPath - ExcitationFilterRef	2	0	0	182
Line - FillColor	0	0	0	184
Line - FillRule	0	0	0	184
Line - FontFamily	0	0	0	184
Line - FontSize	3	0	0	181
Line - FontStyle	0	0	0	184
Line - ID	8	0	0	176
Line - Locked	0	0	0	184
Line - MarkerEnd	0	0	0	184
Line - MarkerStart	0	0	0	184
Line - StrokeColor	1	0	0	183
Line - StrokeDashArray	0	0	0	184
Line - StrokeWidth	2	0	0	182
Line - Text	3	0	0	181
Line - TheC	0	0	0	184
Line - TheT	1	0	0	183

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsupported	Partial	Unknown/Missing
Line - TheZ	1	0	0	183
Line - Transform	1	0	0	183
Line - X1	8	0	0	176
Line - X2	8	0	0	176
Line - Y1	8	0	0	176
Line - Y2	8	0	0	176
ListAnnotation - AnnotationRef	0	0	0	184
ListAnnotation - Description	0	0	0	184
ListAnnotation - ID	0	0	0	184
ListAnnotation - Namespace	0	0	0	184
LongAnnotation - AnnotationRef	0	0	0	184
LongAnnotation - Description	0	0	0	184
LongAnnotation - ID	1	0	0	183
LongAnnotation - Namespace	1	0	0	183
LongAnnotation - Value	1	0	0	183
Mask - BinData	3	0	0	181
Mask - BinDataBigEndian	1	0	0	183
Mask - BinDataBigLength	0	0	0	184
Mask - BinDataCompression	0	0	0	184
Mask - FillColor	1	0	0	183
Mask - FillRule	0	0	0	184
Mask - FontFamily	0	0	0	184
Mask - FontSize	0	0	0	184
Mask - Height	3	0	0	181
Mask - ID	3	0	0	181
Mask - Locked	0	0	0	184
Mask - StrokeColor	1	0	0	183
Mask - StrokeDashArray	0	0	0	184
Mask - StrokeWidth	0	0	0	184
Mask - Text	0	0	0	184
Mask - TheC	0	0	0	184
Mask - TheT	0	0	0	184
Mask - TheZ	0	0	0	184
Mask - Transform	0	0	0	184
Mask - Width	3	0	0	181
Mask - X	3	0	0	181
Mask - Y	3	0	0	181
MicrobeamManipulation - ExperimenterRef	0	0	0	184
MicrobeamManipulation - ID	0	0	0	184
MicrobeamManipulation - ROIRef	0	0	0	184
MicrobeamManipulation - Type	0	0	0	184
MicrobeamManipulationLightSourceSettings - Attenuation	0	0	0	184
MicrobeamManipulationLightSourceSettings - ID	0	0	0	184
MicrobeamManipulationLightSourceSettings - Wavelength	0	0	0	184
Microscope - LotNumber	1	0	0	183
Microscope - Manufacturer	2	0	0	182

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsupported	Partial	Unknown/Missing
Microscope - Model	13	0	0	171
Microscope - SerialNumber	4	0	0	180
Microscope - Type	3	0	0	181
Objective - AnnotationRef	0	0	0	184
Objective - CalibratedMagnification	8	0	0	176
Objective - Correction	27	0	0	157
Objective - ID	44	0	0	140
Objective - Immersion	30	0	0	154
Objective - Iris	2	0	0	182
Objective - LensNA	25	0	0	159
Objective - LotNumber	1	0	0	183
Objective - Manufacturer	6	0	0	178
Objective - Model	19	0	0	165
Objective - NominalMagnification	36	0	0	148
Objective - SerialNumber	3	0	0	181
Objective - WorkingDistance	12	0	0	172
ObjectiveSettings - CorrectionCollar	1	0	0	183
ObjectiveSettings - ID	40	0	0	144
ObjectiveSettings - Medium	1	0	0	183
ObjectiveSettings - RefractiveIndex	10	0	0	174
Pixels - AnnotationRef	0	0	0	184
Pixels - BigEndian	184	0	0	0
Pixels - DimensionOrder	184	0	0	0
Pixels - ID	184	0	0	0
Pixels - Interleaved	184	0	0	0
Pixels - PhysicalSizeX	99	0	0	85
Pixels - PhysicalSizeY	99	0	0	85
Pixels - PhysicalSizeZ	50	0	0	134
Pixels - SignificantBits	184	0	0	0
Pixels - SizeC	184	0	0	0
Pixels - SizeT	184	0	0	0
Pixels - SizeX	184	0	0	0
Pixels - SizeY	184	0	0	0
Pixels - SizeZ	184	0	0	0
Pixels - TimeIncrement	17	0	0	167
Pixels - Type	184	0	0	0
Plane - AnnotationRef	0	0	0	184
Plane - DeltaT	33	0	0	151
Plane - ExposureTime	41	0	0	143
Plane - HashSHA1	0	0	0	184
Plane - PositionX	42	0	0	142
Plane - PositionY	42	0	0	142
Plane - PositionZ	35	0	0	149
Plane - TheC	184	0	0	0
Plane - TheT	184	0	0	0
Plane - TheZ	184	0	0	0
Plate - AnnotationRef	0	0	0	184
Plate - ColumnNamingConvention	9	0	0	175
Plate - Columns	16	0	0	168

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsupported	Partial	Unknown/Missing
Plate - Description	3	0	0	181
Plate - ExternalIdentifier	4	0	0	180
Plate - ID	16	0	0	168
Plate - Name	14	0	0	170
Plate - RowNamingConvention	9	0	0	175
Plate - Rows	16	0	0	168
Plate - Status	0	0	0	184
Plate - WellOriginX	1	0	0	183
Plate - WellOriginY	1	0	0	183
PlateAcquisition - AnnotationRef	0	0	0	184
PlateAcquisition - Description	0	0	0	184
PlateAcquisition - EndTime	3	0	0	181
PlateAcquisition - ID	12	0	0	172
PlateAcquisition - MaximumFieldCount	10	0	0	174
PlateAcquisition - Name	0	0	0	184
PlateAcquisition - StartTime	6	0	0	178
PlateAcquisition - WellSampleRef	11	0	0	173
Point - FillColor	0	0	0	184
Point - FillRule	0	0	0	184
Point - FontFamily	0	0	0	184
Point - FontSize	1	0	0	183
Point - FontStyle	0	0	0	184
Point - ID	6	0	0	178
Point - Locked	0	0	0	184
Point - StrokeColor	1	0	0	183
Point - StrokeDashArray	1	0	0	183
Point - StrokeWidth	2	0	0	182
Point - Text	1	0	0	183
Point - TheC	0	0	0	184
Point - TheT	1	0	0	183
Point - TheZ	2	0	0	182
Point - Transform	0	0	0	184
Point - X	6	0	0	178
Point - Y	6	0	0	178
Polygon - FillColor	0	0	0	184
Polygon - FillRule	0	0	0	184
Polygon - FontFamily	0	0	0	184
Polygon - FontSize	2	0	0	182
Polygon - FontStyle	0	0	0	184
Polygon - ID	9	0	0	175
Polygon - Locked	0	0	0	184
Polygon - Points	9	0	0	175
Polygon - StrokeColor	1	0	0	183
Polygon - StrokeDashArray	1	0	0	183
Polygon - StrokeWidth	3	0	0	181
Polygon - Text	2	0	0	182
Polygon - TheC	0	0	0	184
Polygon - TheT	1	0	0	183
Polygon - TheZ	2	0	0	182

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsupported	Partial	Unknown/Missing
Polygon - Transform	1	0	0	183
Polyline - FillColor	0	0	0	184
Polyline - FillRule	0	0	0	184
Polyline - FontFamily	0	0	0	184
Polyline - FontSize	2	0	0	182
Polyline - FontStyle	0	0	0	184
Polyline - ID	6	0	0	178
Polyline - Locked	0	0	0	184
Polyline - MarkerEnd	0	0	0	184
Polyline - MarkerStart	0	0	0	184
Polyline - Points	6	0	0	178
Polyline - StrokeColor	1	0	0	183
Polyline - StrokeDashArray	1	0	0	183
Polyline - StrokeWidth	3	0	0	181
Polyline - Text	2	0	0	182
Polyline - TheC	0	0	0	184
Polyline - TheT	1	0	0	183
Polyline - TheZ	2	0	0	182
Polyline - Transform	1	0	0	183
Project - AnnotationRef	0	0	0	184
Project - DatasetRef	0	0	0	184
Project - Description	0	0	0	184
Project - ExperimenterGroupRef	0	0	0	184
Project - ExperimenterRef	0	0	0	184
Project - ID	0	0	0	184
Project - Name	0	0	0	184
ROI - AnnotationRef	0	0	0	184
ROI - Description	1	0	0	183
ROI - ID	17	0	0	167
ROI - Name	6	0	0	178
Reagent - AnnotationRef	0	0	0	184
Reagent - Description	0	0	0	184
Reagent - ID	0	0	0	184
Reagent - Name	0	0	0	184
Reagent - ReagentIdentifier	0	0	0	184
Rectangle - FillColor	0	0	0	184
Rectangle - FillRule	0	0	0	184
Rectangle - FontFamily	0	0	0	184
Rectangle - FontSize	3	0	0	181
Rectangle - FontStyle	0	0	0	184
Rectangle - Height	13	0	0	171
Rectangle - ID	13	0	0	171
Rectangle - Locked	0	0	0	184
Rectangle - StrokeColor	3	0	0	181
Rectangle - StrokeDashArray	0	0	0	184
Rectangle - StrokeWidth	2	0	0	182
Rectangle - Text	6	0	0	178
Rectangle - TheC	3	0	0	181
Rectangle - TheT	4	0	0	180

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsupported	Partial	Unknown/Missing
Rectangle - TheZ	4	0	0	180
Rectangle - Transform	1	0	0	183
Rectangle - Width	13	0	0	171
Rectangle - X	13	0	0	171
Rectangle - Y	13	0	0	171
Screen - AnnotationRef	0	0	0	184
Screen - Description	0	0	0	184
Screen - ID	1	0	0	183
Screen - Name	1	0	0	183
Screen - PlateRef	0	0	0	184
Screen - ProtocolDescription	0	0	0	184
Screen - ProtocolIdentifier	0	0	0	184
Screen - ReagentSetDescription	0	0	0	184
Screen - ReagentSetIdentifier	0	0	0	184
Screen - Type	0	0	0	184
StageLabel - Name	3	0	0	181
StageLabel - X	2	0	0	182
StageLabel - Y	2	0	0	182
StageLabel - Z	3	0	0	181
TagAnnotation - AnnotationRef	0	0	0	184
TagAnnotation - Description	0	0	0	184
TagAnnotation - ID	1	0	0	183
TagAnnotation - Namespace	1	0	0	183
TagAnnotation - Value	1	0	0	183
TermAnnotation - AnnotationRef	0	0	0	184
TermAnnotation - Description	0	0	0	184
TermAnnotation - ID	1	0	0	183
TermAnnotation - Namespace	1	0	0	183
TermAnnotation - Value	1	0	0	183
TiffData - FirstC	0	0	0	184
TiffData - FirstT	0	0	0	184
TiffData - FirstZ	0	0	0	184
TiffData - IFD	0	0	0	184
TiffData - PlaneCount	0	0	0	184
TimestampAnnotation - AnnotationRef	0	0	0	184
TimestampAnnotation - Description	0	0	0	184
TimestampAnnotation - ID	1	0	0	183
TimestampAnnotation - Namespace	1	0	0	183
TimestampAnnotation - Value	1	0	0	183
TransmittanceRange - CutIn	5	0	0	179
TransmittanceRange - CutInTolerance	1	0	0	183
TransmittanceRange - CutOut	5	0	0	179
TransmittanceRange - CutOutTolerance	1	0	0	183
TransmittanceRange - Transmittance	1	0	0	183
UUID - FileName	0	0	0	184
UUID - Value	0	0	0	184
Well - AnnotationRef	0	0	0	184
Well - Color	0	0	0	184
Well - Column	16	0	0	168

continues on next page

Table 4 – continued from previous page

Field	Supported	Unsupported	Partial	Unknown/Missing
Well - ExternalDescription	0	0	0	184
Well - ExternalIdentifier	1	0	0	183
Well - ID	16	0	0	168
Well - ReagentRef	0	0	0	184
Well - Row	16	0	0	168
Well - Type	0	0	0	184
WellSample - AnnotationRef	0	0	0	184
WellSample - ID	16	0	0	168
WellSample - ImageRef	16	0	0	168
WellSample - Index	16	0	0	168
WellSample - PositionX	10	0	0	174
WellSample - PositionY	10	0	0	174
WellSample - Timepoint	0	0	0	184
XMLAnnotation - AnnotationRef	0	0	0	184
XMLAnnotation - ID	2	0	0	182
XMLAnnotation - Namespace	1	0	0	183
XMLAnnotation - Value	2	0	0	182

AFIReader

This page lists supported metadata fields for the Bio-Formats Aperio AFI format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 31 of them (6%).
- Of those, Bio-Formats fully or partially converts 31 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Aperio AFI format reader:

- Channel : Color
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID

- Objective : ID
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 31

Total unknown or missing: 445

AIMReader

This page lists supported metadata fields for the Bio-Formats AIM format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats AIM format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID

- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

APLReader

This page lists supported metadata fields for the Bio-Formats Olympus APL format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 21 of them (4%).
- Of those, Bio-Formats fully or partially converts 21 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Olympus APL format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian

- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 21

Total unknown or missing: 455

APNGReader

This page lists supported metadata fields for the Bio-Formats Animated PNG format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Animated PNG format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved

- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

ARFReader

This page lists supported metadata fields for the Bio-Formats ARF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats ARF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY

- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

AVIReader

This page lists supported metadata fields for the Bio-Formats Audio Video Interleave format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Audio Video Interleave format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

AliconaReader

This page lists supported metadata fields for the Bio-Formats Alicona AL3D format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 33 of them (6%).
- Of those, Bio-Formats fully or partially converts 33 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Alicona AL3D format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Type
- DetectorSettings : ID
- DetectorSettings : Voltage
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : CalibratedMagnification
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : WorkingDistance
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits

- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 33

Total unknown or missing: 443

AmiraReader

This page lists supported metadata fields for the Bio-Formats Amira format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Amira format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT

- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

AnalyzeReader

This page lists supported metadata fields for the Bio-Formats Analyze 7.5 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 24 of them (5%).
- Of those, Bio-Formats fully or partially converts 24 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Analyze 7.5 format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX

- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 24

Total unknown or missing: 452

BDReader

This page lists supported metadata fields for the Bio-Formats BD Pathway format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 59 of them (12%).
- Of those, Bio-Formats fully or partially converts 59 (100%).

Supported fields

These fields are fully supported by the Bio-Formats BD Pathway format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : Offset
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- Instrument : ID
- Objective : ID

- Objective : LensNA
- Objective : Manufacturer
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ColumnNamingConvention
- Plate : Columns
- Plate : Description
- Plate : ID
- Plate : Name
- Plate : RowNamingConvention
- Plate : Rows
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : WellSampleRef
- ROI : ID
- Rectangle : Height
- Rectangle : ID
- Rectangle : Width
- Rectangle : X
- Rectangle : Y

- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index

Total supported: 59

Total unknown or missing: 417

BDVReader

This page lists supported metadata fields for the Bio-Formats BDV format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 37 of them (7%).
- Of those, Bio-Formats fully or partially converts 37 (100%).

Supported fields

These fields are fully supported by the Bio-Formats BDV format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Image : ROIRef
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY

- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- ROI : ID
- ROI : Name
- Rectangle : Height
- Rectangle : ID
- Rectangle : StrokeColor
- Rectangle : Text
- Rectangle : TheC
- Rectangle : TheT
- Rectangle : TheZ
- Rectangle : Width
- Rectangle : X
- Rectangle : Y
- XMLAnnotation : ID
- XMLAnnotation : Value

Total supported: 37

Total unknown or missing: 439

BIFormatReader

This page lists supported metadata fields for the Bio-Formats BIFormatReader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats BIFormatReader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name

- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

BMPReader

This page lists supported metadata fields for the Bio-Formats Windows Bitmap format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 21 of them (4%).
- Of those, Bio-Formats fully or partially converts 21 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Windows Bitmap format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX

- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 21

Total unknown or missing: 455

BaseTiffReader

This page lists supported metadata fields for the Bio-Formats BaseTiffReader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 28 of them (5%).
- Of those, Bio-Formats fully or partially converts 28 (100%).

Supported fields

These fields are fully supported by the Bio-Formats BaseTiffReader:

- Channel : ID
- Channel : SamplesPerPixel
- Experimenter : Email
- Experimenter : FirstName
- Experimenter : ID
- Experimenter : LastName
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID

- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 28

Total unknown or missing: 448

BaseZeissReader

This page lists supported metadata fields for the Bio-Formats BaseZeissReader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 84 of them (17%).
- Of those, Bio-Formats fully or partially converts 84 (100%).

Supported fields

These fields are fully supported by the Bio-Formats BaseZeissReader:

- Channel : Color
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Type
- DetectorSettings : Gain

- DetectorSettings : ID
- DetectorSettings : Offset
- Ellipse : ID
- Ellipse : RadiusX
- Ellipse : RadiusY
- Ellipse : Text
- Ellipse : X
- Ellipse : Y
- Experimenter : FirstName
- Experimenter : ID
- Experimenter : Institution
- Experimenter : LastName
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- Instrument : ID
- Label : ID
- Label : Text
- Label : X
- Label : Y
- Line : ID
- Line : Text
- Line : X1
- Line : X2
- Line : Y1
- Line : Y2
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : NominalMagnification
- Objective : WorkingDistance
- ObjectiveSettings : ID

- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Point : ID
- Point : Text
- Point : X
- Point : Y
- Polygon : ID
- Polygon : Points
- Polygon : Text
- Polyline : ID
- Polyline : Points
- Polyline : Text
- ROI : ID
- ROI : Name
- Rectangle : Height
- Rectangle : ID
- Rectangle : Text

- Rectangle : Width
- Rectangle : X
- Rectangle : Y

Total supported: 84

Total unknown or missing: 392

BioRadGelReader

This page lists supported metadata fields for the Bio-Formats Bio-Rad GEL format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 21 of them (4%).
- Of those, Bio-Formats fully or partially converts 21 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Bio-Rad GEL format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 21

Total unknown or missing: 455

BioRadReader

This page lists supported metadata fields for the Bio-Formats Bio-Rad PIC format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 40 of them (8%).
- Of those, Bio-Formats fully or partially converts 40 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Bio-Rad PIC format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : Gain
- Detector : ID
- Detector : Offset
- Detector : Type
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : Offset
- Experiment : ID
- Experiment : Type
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Model
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian

- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 40

Total unknown or missing: 436

BioRadSCNReader

This page lists supported metadata fields for the Bio-Formats Bio-Rad SCN format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 29 of them (6%).
- Of those, Bio-Formats fully or partially converts 29 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Bio-Rad SCN format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- Image : AcquisitionDate
- Image : ID

- Image : Name
- Instrument : ID
- Microscope : Model
- Microscope : SerialNumber
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 29

Total unknown or missing: 447

BrukerReader

This page lists supported metadata fields for the Bio-Formats Bruker format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 23 of them (4%).
- Of those, Bio-Formats fully or partially converts 23 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Bruker format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Experimenter : ID

- Experimenter : Institution
- Experimenter : LastName
- Image : AcquisitionDate
- Image : ExperimenterRef
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 23

Total unknown or missing: 453

BurleighReader

This page lists supported metadata fields for the Bio-Formats Burleigh format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Burleigh format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID

- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

CV7000Reader

This page lists supported metadata fields for the Bio-Formats Yokogawa CV7000 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 57 of them (11%).
- Of those, Bio-Formats fully or partially converts 57 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Yokogawa CV7000 format reader:

- Channel : Color
- Channel : ExcitationWavelength
- Channel : Fluor
- Channel : ID
- Channel : LightSourceSettingsID
- Channel : Name

- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Instrument : ID
- Laser : ID
- Laser : Power
- Laser : Wavelength
- Objective : ID
- Objective : Model
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : Columns
- Plate : Description
- Plate : ExternalIdentifier
- Plate : ID

- Plate : Name
- Plate : Rows
- PlateAcquisition : EndTime
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : StartTime
- PlateAcquisition : WellSampleRef
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index
- WellSample : PositionX
- WellSample : PositionY

Total supported: 57

Total unknown or missing: 419

CanonRawReader

This page lists supported metadata fields for the Bio-Formats Canon RAW format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Canon RAW format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved

- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

CellH5Reader

This page lists supported metadata fields for the Bio-Formats CellH5 (HDF) format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 41 of them (8%).
- Of those, Bio-Formats fully or partially converts 41 (100%).

Supported fields

These fields are fully supported by the Bio-Formats CellH5 (HDF) format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Image : ROIRef
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX

- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ID
- Plate : Name
- ROI : ID
- ROI : Name
- Rectangle : Height
- Rectangle : ID
- Rectangle : StrokeColor
- Rectangle : Text
- Rectangle : TheC
- Rectangle : TheT
- Rectangle : TheZ
- Rectangle : Width
- Rectangle : X
- Rectangle : Y
- Well : Column
- Well : ExternalIdentifier
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index

Total supported: 41

Total unknown or missing: 435

CellSensReader

This page lists supported metadata fields for the Bio-Formats CellSens VSI format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 49 of them (10%).

- Of those, Bio-Formats fully or partially converts 49 (100%).

Supported fields

These fields are fully supported by the Bio-Formats CellSens VSI format reader:

- Channel : EmissionWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : Gain
- Detector : ID
- Detector : Manufacturer
- Detector : Model
- Detector : Offset
- Detector : SerialNumber
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : Offset
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : ID
- Objective : LensNA
- Objective : Model
- Objective : NominalMagnification
- Objective : WorkingDistance
- ObjectiveSettings : ID
- ObjectiveSettings : RefractiveIndex
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX

- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 49

Total unknown or missing: 427

CellVoyagerReader

This page lists supported metadata fields for the Bio-Formats CellVoyager format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 39 of them (8%).
- Of those, Bio-Formats fully or partially converts 39 (100%).

Supported fields

These fields are fully supported by the Bio-Formats CellVoyager format reader:

- Channel : Color
- Channel : ID
- Channel : Name
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID

- Image : InstrumentRef
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : Columns
- Plate : Name
- Plate : Rows
- PlateAcquisition : EndTime
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : StartTime
- PlateAcquisition : WellSampleRef
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index
- WellSample : PositionX
- WellSample : PositionY

Total supported: 39

Total unknown or missing: 437

CellWorxReader

This page lists supported metadata fields for the Bio-Formats CellWorx format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 50 of them (10%).
- Of those, Bio-Formats fully or partially converts 50 (100%).

Supported fields

These fields are fully supported by the Bio-Formats CellWorx format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- DetectorSettings : Gain
- DetectorSettings : ID
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Microscope : SerialNumber
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ

- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : Columns
- Plate : ID
- Plate : Name
- Plate : Rows
- PlateAcquisition : EndTime
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : StartTime
- PlateAcquisition : WellSampleRef
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index
- WellSample : PositionX
- WellSample : PositionY

Total supported: 50

Total unknown or missing: 426

CellomicsReader

This page lists supported metadata fields for the Bio-Formats Cellomics C01 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 36 of them (7%).
- Of those, Bio-Formats fully or partially converts 36 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Cellomics C01 format reader:

- Channel : Color
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ColumnNamingConvention
- Plate : Columns
- Plate : ID
- Plate : Name
- Plate : RowNamingConvention
- Plate : Rows
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID

- WellSample : ImageRef
- WellSample : Index

Total supported: 36

Total unknown or missing: 440

ColumbusReader

This page lists supported metadata fields for the Bio-Formats PerkinElmer Columbus format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 43 of them (9%).
- Of those, Bio-Formats fully or partially converts 43 (100%).

Supported fields

These fields are fully supported by the Bio-Formats PerkinElmer Columbus format reader:

- Channel : Color
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type

- Plane : DeltaT
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : Columns
- Plate : ID
- Plate : Name
- Plate : Rows
- Screen : ID
- Screen : Name
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index
- WellSample : PositionX
- WellSample : PositionY

Total supported: 43

Total unknown or missing: 433

DCIMGReader

This page lists supported metadata fields for the Bio-Formats Hamamatsu DCIMG format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Hamamatsu DCIMG format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate

- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

DNGReader

This page lists supported metadata fields for the Bio-Formats DNG format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats DNG format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID

- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

DeltavisionReader

This page lists supported metadata fields for the Bio-Formats Deltavision format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 51 of them (10%).
- Of those, Bio-Formats fully or partially converts 51 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Deltavision format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : NDFilter
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : ReadOutRate

- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- ImagingEnvironment : Temperature
- Instrument : ID
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Manufacturer
- Objective : Model
- Objective : NominalMagnification
- Objective : WorkingDistance
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC

- Plane : TheT
- Plane : TheZ

Total supported: 51

Total unknown or missing: 425

DicomReader

This page lists supported metadata fields for the Bio-Formats DICOM format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 27 of them (5%).
- Of those, Bio-Formats fully or partially converts 27 (100%).

Supported fields

These fields are fully supported by the Bio-Formats DICOM format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX

- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 27

Total unknown or missing: 449

EPSReader

This page lists supported metadata fields for the Bio-Formats Encapsulated PostScript format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Encapsulated PostScript format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

Ecat7Reader

This page lists supported metadata fields for the Bio-Formats ECAT7 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 23 of them (4%).
- Of those, Bio-Formats fully or partially converts 23 (100%).

Supported fields

These fields are fully supported by the Bio-Formats ECAT7 format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 23

Total unknown or missing: 453

FEIReader

This page lists supported metadata fields for the Bio-Formats FEI/Philips format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats FEI/Philips format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

FEITiffReader

This page lists supported metadata fields for the Bio-Formats FEI TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 39 of them (8%).
- Of those, Bio-Formats fully or partially converts 39 (100%).

Supported fields

These fields are fully supported by the Bio-Formats FEI TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- Detector : Type
- Experimenter : ID
- Experimenter : LastName
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Microscope : Model
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : NominalMagnification
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC

- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- StageLabel : Name
- StageLabel : X
- StageLabel : Y
- StageLabel : Z

Total supported: 39

Total unknown or missing: 437

FV1000Reader

This page lists supported metadata fields for the Bio-Formats Olympus FV1000 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 113 of them (23%).
- Of those, Bio-Formats fully or partially converts 113 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Olympus FV1000 format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : IlluminationType
- Channel : LightSourceSettingsID
- Channel : LightSourceSettingsWavelength
- Channel : Name
- Channel : SamplesPerPixel
- Detector : Gain
- Detector : ID
- Detector : Type

- Detector : Voltage
- DetectorSettings : ID
- Dichroic : ID
- Dichroic : Model
- Ellipse : FontSize
- Ellipse : ID
- Ellipse : RadiusX
- Ellipse : RadiusY
- Ellipse : StrokeWidth
- Ellipse : TheT
- Ellipse : TheZ
- Ellipse : Transform
- Ellipse : X
- Ellipse : Y
- Filter : ID
- Filter : Model
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- Instrument : ID
- Laser : ID
- Laser : LaserMedium
- Laser : Type
- Laser : Wavelength
- LightPath : DichroicRef
- LightPath : EmissionFilterRef
- Line : FontSize
- Line : ID
- Line : StrokeWidth
- Line : TheT
- Line : TheZ
- Line : Transform
- Line : X1
- Line : X2

- Line : Y1
- Line : Y2
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Model
- Objective : NominalMagnification
- Objective : WorkingDistance
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Point : FontSize
- Point : ID
- Point : StrokeWidth
- Point : TheT

- Point : TheZ
- Point : X
- Point : Y
- Polygon : FontSize
- Polygon : ID
- Polygon : Points
- Polygon : StrokeWidth
- Polygon : TheT
- Polygon : TheZ
- Polygon : Transform
- Polyline : FontSize
- Polyline : ID
- Polyline : Points
- Polyline : StrokeWidth
- Polyline : TheT
- Polyline : TheZ
- Polyline : Transform
- ROI : ID
- Rectangle : FontSize
- Rectangle : Height
- Rectangle : ID
- Rectangle : StrokeWidth
- Rectangle : TheT
- Rectangle : TheZ
- Rectangle : Transform
- Rectangle : Width
- Rectangle : X
- Rectangle : Y
- TransmittanceRange : CutIn
- TransmittanceRange : CutOut

Total supported: 113

Total unknown or missing: 363

FakeReader

This page lists supported metadata fields for the Bio-Formats Simulated data format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 101 of them (21%).
- Of those, Bio-Formats fully or partially converts 101 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Simulated data format reader:

- BooleanAnnotation : ID
- BooleanAnnotation : Namespace
- BooleanAnnotation : Value
- Channel : Color
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : FilterSetRef
- Channel : ID
- Channel : LightSourceSettingsID
- Channel : Name
- Channel : SamplesPerPixel
- CommentAnnotation : ID
- CommentAnnotation : Namespace
- CommentAnnotation : Value
- DetectorSettings : ID
- DoubleAnnotation : ID
- DoubleAnnotation : Namespace
- DoubleAnnotation : Value
- Ellipse : ID
- Ellipse : RadiusX
- Ellipse : RadiusY
- Ellipse : X
- Ellipse : Y
- Image : AcquisitionDate
- Image : AnnotationRef
- Image : ID

- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- Label : ID
- Label : Text
- Label : X
- Label : Y
- LightPath : DichroicRef
- LightPath : EmissionFilterRef
- LightPath : ExcitationFilterRef
- Line : ID
- Line : X1
- Line : X2
- Line : Y1
- Line : Y2
- LongAnnotation : ID
- LongAnnotation : Namespace
- LongAnnotation : Value
- Mask : BinData
- Mask : BinDataBigEndian
- Mask : Height
- Mask : ID
- Mask : Width
- Mask : X
- Mask : Y
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT

- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Point : ID
- Point : X
- Point : Y
- Polygon : ID
- Polygon : Points
- Polyline : ID
- Polyline : Points
- ROI : ID
- Rectangle : Height
- Rectangle : ID
- Rectangle : Width
- Rectangle : X
- Rectangle : Y
- TagAnnotation : ID
- TagAnnotation : Namespace
- TagAnnotation : Value
- TermAnnotation : ID
- TermAnnotation : Namespace
- TermAnnotation : Value
- TimestampAnnotation : ID
- TimestampAnnotation : Namespace
- TimestampAnnotation : Value
- WellSample : PositionX
- WellSample : PositionY

- XMLAnnotation : ID
- XMLAnnotation : Namespace
- XMLAnnotation : Value

Total supported: 101

Total unknown or missing: 375

FilePatternReader

This page lists supported metadata fields for the Bio-Formats File pattern format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 20 of them (4%).
- Of those, Bio-Formats fully or partially converts 20 (100%).

Supported fields

These fields are fully supported by the Bio-Formats File pattern format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 20

Total unknown or missing: 456

FitsReader

This page lists supported metadata fields for the Bio-Formats Flexible Image Transport System format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Flexible Image Transport System format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

FlexReader

This page lists supported metadata fields for the Bio-Formats Evotec Flex format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 71 of them (14%).
- Of those, Bio-Formats fully or partially converts 71 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Evotec Flex format reader:

- Channel : ID
- Channel : LightSourceSettingsID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : ID
- Dichroic : ID
- Dichroic : Model
- Filter : FilterWheel
- Filter : ID
- Filter : Model
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Laser : ID
- Laser : LaserMedium
- Laser : Type
- Laser : Wavelength
- LightPath : DichroicRef
- LightPath : EmissionFilterRef
- LightPath : ExcitationFilterRef
- Objective : CalibratedMagnification

- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ColumnNamingConvention
- Plate : Columns
- Plate : ExternalIdentifier
- Plate : ID
- Plate : Name
- Plate : RowNamingConvention
- Plate : Rows
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : StartTime

- PlateAcquisition : WellSampleRef
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index
- WellSample : PositionX
- WellSample : PositionY

Total supported: 71

Total unknown or missing: 405

FlowSightReader

This page lists supported metadata fields for the Bio-Formats FlowSight format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 20 of them (4%).
- Of those, Bio-Formats fully or partially converts 20 (100%).

Supported fields

These fields are fully supported by the Bio-Formats FlowSight format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY

- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 20

Total unknown or missing: 456

FluoviewReader

This page lists supported metadata fields for the Bio-Formats Olympus Fluoview/ABD TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 49 of them (10%).
- Of those, Bio-Formats fully or partially converts 49 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Olympus Fluoview/ABD TIFF format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Manufacturer
- Detector : Model
- Detector : Type
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : Offset
- DetectorSettings : ReadOutRate
- DetectorSettings : Voltage
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- ImagingEnvironment : Temperature
- Instrument : ID

- Objective : CalibratedMagnification
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Model
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 49

Total unknown or missing: 427

FujiReader

This page lists supported metadata fields for the Bio-Formats Fuji LAS 3000 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 23 of them (4%).
- Of those, Bio-Formats fully or partially converts 23 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Fuji LAS 3000 format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Instrument : ID
- Microscope : Model
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 23

Total unknown or missing: 453

GIFReader

This page lists supported metadata fields for the Bio-Formats Graphics Interchange Format format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Graphics Interchange Format format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

GatanDM2Reader

This page lists supported metadata fields for the Bio-Formats Gatan DM2 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 30 of them (6%).
- Of those, Bio-Formats fully or partially converts 30 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Gatan DM2 format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- DetectorSettings : Binning
- DetectorSettings : ID
- Experimenter : FirstName
- Experimenter : ID
- Experimenter : LastName
- Image : AcquisitionDate
- Image : ExperimenterRef
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 30

Total unknown or missing: 446

GatanReader

This page lists supported metadata fields for the Bio-Formats Gatan Digital Micrograph format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 69 of them (14%).
- Of those, Bio-Formats fully or partially converts 69 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Gatan Digital Micrograph format reader:

- Channel : AcquisitionMode
- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- DetectorSettings : ID
- DetectorSettings : Voltage
- Ellipse : FontSize
- Ellipse : ID
- Ellipse : RadiusX
- Ellipse : RadiusY
- Ellipse : StrokeColor
- Ellipse : Text
- Ellipse : X
- Ellipse : Y
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- Instrument : ID
- Label : FontSize
- Label : ID
- Label : StrokeColor
- Label : Text

- Label : X
- Label : Y
- Line : FontSize
- Line : ID
- Line : StrokeColor
- Line : Text
- Line : X1
- Line : X2
- Line : Y1
- Line : Y2
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

- ROI : ID
- Rectangle : FontSize
- Rectangle : Height
- Rectangle : ID
- Rectangle : StrokeColor
- Rectangle : Text
- Rectangle : Width
- Rectangle : X
- Rectangle : Y

Total supported: 69

Total unknown or missing: 407

GelReader

This page lists supported metadata fields for the Bio-Formats Amersham Biosciences GEL format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 21 of them (4%).
- Of those, Bio-Formats fully or partially converts 21 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Amersham Biosciences GEL format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX

- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 21

Total unknown or missing: 455

HISReader

This page lists supported metadata fields for the Bio-Formats Hamamatsu HIS format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 27 of them (5%).
- Of those, Bio-Formats fully or partially converts 27 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Hamamatsu HIS format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Offset
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : ID
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC

- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 27

Total unknown or missing: 449

HRDGDFReader

This page lists supported metadata fields for the Bio-Formats NOAA-HRD Gridded Data Format format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 21 of them (4%).
- Of those, Bio-Formats fully or partially converts 21 (100%).

Supported fields

These fields are fully supported by the Bio-Formats NOAA-HRD Gridded Data Format format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX

- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 21

Total unknown or missing: 455

HamamatsuVMSReader

This page lists supported metadata fields for the Bio-Formats Hamamatsu VMS format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 26 of them (5%).
- Of those, Bio-Formats fully or partially converts 26 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Hamamatsu VMS format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : ID
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC

- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 26

Total unknown or missing: 450

HitachiReader

This page lists supported metadata fields for the Bio-Formats Hitachi format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 31 of them (6%).
- Of those, Bio-Formats fully or partially converts 31 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Hitachi format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Microscope : Model
- Microscope : SerialNumber
- Objective : ID
- Objective : WorkingDistance
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved

- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 31

Total unknown or missing: 445

I2IReader

This page lists supported metadata fields for the Bio-Formats I2I format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats I2I format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved

- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

ICSReader

This page lists supported metadata fields for the Bio-Formats Image Cytometry Standard format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 72 of them (15%).
- Of those, Bio-Formats fully or partially converts 72 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Image Cytometry Standard format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Manufacturer
- Detector : Model
- Detector : Type
- DetectorSettings : Gain
- DetectorSettings : ID
- Dichroic : ID
- Dichroic : Model

- Experiment : ID
- Experiment : Type
- Experimenter : ID
- Experimenter : LastName
- Filter : ID
- Filter : Model
- FilterSet : DichroicRef
- FilterSet : EmissionFilterRef
- FilterSet : ExcitationFilterRef
- FilterSet : ID
- FilterSet : Model
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Laser : ID
- Laser : LaserMedium
- Laser : Manufacturer
- Laser : Model
- Laser : Power
- Laser : RepetitionRate
- Laser : Type
- Laser : Wavelength
- Microscope : Manufacturer
- Microscope : Model
- Objective : CalibratedMagnification
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Model
- Objective : WorkingDistance
- ObjectiveSettings : ID
- Pixels : BigEndian

- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 72

Total unknown or missing: 404

IM3Reader

This page lists supported metadata fields for the Bio-Formats Perkin-Elmer Nuance IM3 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Perkin-Elmer Nuance IM3 format reader:

- Channel : ID
- Channel : SamplesPerPixel

- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

IMODReader

This page lists supported metadata fields for the Bio-Formats IMOD format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 44 of them (9%).
- Of those, Bio-Formats fully or partially converts 44 (100%).

Supported fields

These fields are fully supported by the Bio-Formats IMOD format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Image : ROIRef
- Pixels : BigEndian

- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Point : ID
- Point : StrokeColor
- Point : StrokeDashArray
- Point : StrokeWidth
- Point : TheZ
- Point : X
- Point : Y
- Polygon : ID
- Polygon : Points
- Polygon : StrokeColor
- Polygon : StrokeDashArray
- Polygon : StrokeWidth
- Polygon : TheZ
- Polyline : ID
- Polyline : Points
- Polyline : StrokeColor
- Polyline : StrokeDashArray
- Polyline : StrokeWidth
- Polyline : TheZ
- ROI : ID

- ROI : Name

Total supported: 44

Total unknown or missing: 432

INRReader

This page lists supported metadata fields for the Bio-Formats INR format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats INR format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

IPLabReader

This page lists supported metadata fields for the Bio-Formats IPLab format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 31 of them (6%).
- Of those, Bio-Formats fully or partially converts 31 (100%).

Supported fields

These fields are fully supported by the Bio-Formats IPLab format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Image : ROIRef
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : TheC
- Plane : TheT

- Plane : TheZ
- ROI : ID
- Rectangle : Height
- Rectangle : ID
- Rectangle : Width
- Rectangle : X
- Rectangle : Y

Total supported: 31

Total unknown or missing: 445

IPWReader

This page lists supported metadata fields for the Bio-Formats Image-Pro Workspace format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 20 of them (4%).
- Of those, Bio-Formats fully or partially converts 20 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Image-Pro Workspace format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type

- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 20

Total unknown or missing: 456

ImaconReader

This page lists supported metadata fields for the Bio-Formats Imacon format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 23 of them (4%).
- Of those, Bio-Formats fully or partially converts 23 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Imacon format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Experimenter : FirstName
- Experimenter : ID
- Experimenter : LastName
- Image : AcquisitionDate
- Image : ExperimenterRef
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC

- Plane : TheT
- Plane : TheZ

Total supported: 23

Total unknown or missing: 453

ImageIOReader

This page lists supported metadata fields for the Bio-Formats ImageIOReader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats ImageIOReader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

ImagicReader

This page lists supported metadata fields for the Bio-Formats IMAGIC format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats IMAGIC format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

ImarisHDFReader

This page lists supported metadata fields for the Bio-Formats Bitplane Imaris 5.5 (HDF) format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 24 of them (5%).
- Of those, Bio-Formats fully or partially converts 24 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Bitplane Imaris 5.5 (HDF) format reader:

- Channel : Color
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 24

Total unknown or missing: 452

ImarisReader

This page lists supported metadata fields for the Bio-Formats Bitplane Imaris format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 32 of them (6%).
- Of those, Bio-Formats fully or partially converts 32 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Bitplane Imaris format reader:

- Channel : ID
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Type
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : Offset
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY

- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 32

Total unknown or missing: 444

ImarisTiffReader

This page lists supported metadata fields for the Bio-Formats Bitplane Imaris 3 (TIFF) format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 23 of them (4%).
- Of those, Bio-Formats fully or partially converts 23 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Bitplane Imaris 3 (TIFF) format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY

- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 23

Total unknown or missing: 453

ImprovisionTiffReader

This page lists supported metadata fields for the Bio-Formats Improvision TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 26 of them (5%).
- Of those, Bio-Formats fully or partially converts 26 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Improvision TIFF format reader:

- Channel : Color
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX

- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 26

Total unknown or missing: 450

ImspectorReader

This page lists supported metadata fields for the Bio-Formats Lavision Imspector format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Lavision Imspector format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC

- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

InCell3000Reader

This page lists supported metadata fields for the Bio-Formats InCell 3000 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats InCell 3000 format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

InCellReader

This page lists supported metadata fields for the Bio-Formats InCell 1000/2000 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 69 of them (14%).
- Of those, Bio-Formats fully or partially converts 69 (100%).

Supported fields

These fields are fully supported by the Bio-Formats InCell 1000/2000 format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- Experiment : ID
- Experiment : Type
- Image : AcquisitionDate
- Image : Description
- Image : ExperimentRef
- Image : ID
- Image : InstrumentRef
- Image : Name
- ImagingEnvironment : Temperature
- Instrument : ID
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Manufacturer

- Objective : NominalMagnification
- ObjectiveSettings : ID
- ObjectiveSettings : RefractiveIndex
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ColumnNamingConvention
- Plate : Columns
- Plate : ID
- Plate : Name
- Plate : RowNamingConvention
- Plate : Rows
- Plate : WellOriginX
- Plate : WellOriginY
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : WellSampleRef
- Well : Column

- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index
- WellSample : PositionX
- WellSample : PositionY

Total supported: 69

Total unknown or missing: 407

InveonReader

This page lists supported metadata fields for the Bio-Formats Inveon format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 30 of them (6%).
- Of those, Bio-Formats fully or partially converts 30 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Inveon format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Experimenter : ID
- Experimenter : Institution
- Experimenter : UserName
- Image : AcquisitionDate
- Image : Description
- Image : ExperimenterRef
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Microscope : Model
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved

- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 30

Total unknown or missing: 446

IonpathMIBITiffReader

This page lists supported metadata fields for the Bio-Formats Ionpath MIBI format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Ionpath MIBI format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Instrument : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID

- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

IvisionReader

This page lists supported metadata fields for the Bio-Formats IVision format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 34 of them (7%).
- Of those, Bio-Formats fully or partially converts 34 (100%).

Supported fields

These fields are fully supported by the Bio-Formats IVision format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : Correction

- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : NominalMagnification
- ObjectiveSettings : ID
- ObjectiveSettings : RefractiveIndex
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 34

Total unknown or missing: 442

JDCEReader

This page lists supported metadata fields for the Bio-Formats Molecular Devices JDCE format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 39 of them (8%).
- Of those, Bio-Formats fully or partially converts 39 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Molecular Devices JDCE format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID

- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ColumnNamingConvention
- Plate : Columns
- Plate : ID
- Plate : Name
- Plate : RowNamingConvention
- Plate : Rows
- PlateAcquisition : ID
- PlateAcquisition : StartTime
- WellSample : PositionX
- WellSample : PositionY

Total supported: 39

Total unknown or missing: 437

JEOLReader

This page lists supported metadata fields for the Bio-Formats JEOL format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats JEOL format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

JPEG2000Reader

This page lists supported metadata fields for the Bio-Formats JPEG-2000 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats JPEG-2000 format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

JPEGReader

This page lists supported metadata fields for the Bio-Formats JPEG format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats JPEG format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

JPKReader

This page lists supported metadata fields for the Bio-Formats JPK Instruments format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats JPK Instruments format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

JPXReader

This page lists supported metadata fields for the Bio-Formats JPX format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats JPX format reader:

- Channel : ID
- Channel : SamplesPerPixel

- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

KLBReader

This page lists supported metadata fields for the Bio-Formats KLB format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats KLB format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder

- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

KhorosReader

This page lists supported metadata fields for the Bio-Formats Khoros XV format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Khoros XV format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved

- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

KodakReader

This page lists supported metadata fields for the Bio-Formats Kodak Molecular Imaging format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 26 of them (5%).
- Of those, Bio-Formats fully or partially converts 26 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Kodak Molecular Imaging format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- ImagingEnvironment : Temperature
- Instrument : ID
- Microscope : Model
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX

- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 26

Total unknown or missing: 450

L2DReader

This page lists supported metadata fields for the Bio-Formats Li-Cor L2D format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 29 of them (6%).
- Of those, Bio-Formats fully or partially converts 29 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Li-Cor L2D format reader:

- Channel : ID
- Channel : LightSourceSettingsID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Laser : ID
- Laser : LaserMedium
- Laser : Type

- Laser : Wavelength
- Microscope : Model
- Microscope : Type
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 29

Total unknown or missing: 447

LEOReader

This page lists supported metadata fields for the Bio-Formats LEO format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 27 of them (5%).
- Of those, Bio-Formats fully or partially converts 27 (100%).

Supported fields

These fields are fully supported by the Bio-Formats LEO format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID

- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : WorkingDistance
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 27

Total unknown or missing: 449

LIFReader

This page lists supported metadata fields for the Bio-Formats Leica Image File Format format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 85 of them (17%).
- Of those, Bio-Formats fully or partially converts 85 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Leica Image File Format format reader:

- Channel : Color
- Channel : ExcitationWavelength
- Channel : ID
- Channel : LightSourceSettingsAttenuation

- Channel : LightSourceSettingsID
- Channel : Name
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- Detector : Offset
- Detector : Type
- Detector : Zoom
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : Offset
- Filter : ID
- Filter : Model
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- Instrument : ID
- Label : FontSize
- Label : ID
- Label : StrokeWidth
- Label : Text
- Label : X
- Label : Y
- Laser : ID
- Laser : LaserMedium
- Laser : Type
- Laser : Wavelength
- LightPath : EmissionFilterRef
- Line : ID
- Line : X1
- Line : X2
- Line : Y1

- Line : Y2
- Microscope : Model
- Microscope : Type
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Model
- Objective : NominalMagnification
- Objective : SerialNumber
- ObjectiveSettings : ID
- ObjectiveSettings : RefractiveIndex
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Polygon : ID

- Polygon : Points
- ROI : ID
- Rectangle : Height
- Rectangle : ID
- Rectangle : Width
- Rectangle : X
- Rectangle : Y
- TransmittanceRange : CutIn
- TransmittanceRange : CutOut

Total supported: 85

Total unknown or missing: 391

LIMReader

This page lists supported metadata fields for the Bio-Formats Laboratory Imaging format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Laboratory Imaging format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ

- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

LOFReader

This page lists supported metadata fields for the Bio-Formats Leica Object Format format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Leica Object Format format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

LeicaReader

This page lists supported metadata fields for the Bio-Formats Leica format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 56 of them (11%).
- Of those, Bio-Formats fully or partially converts 56 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Leica format reader:

- Channel : Color
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Offset
- Detector : Type
- Detector : Voltage
- DetectorSettings : ID
- Filter : ID
- Filter : Model
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- LightPath : EmissionFilterRef
- Objective : Correction
- Objective : ID
- Objective : Immersion

- Objective : LensNA
- Objective : Model
- Objective : NominalMagnification
- Objective : SerialNumber
- ObjectiveSettings : ID
- ObjectiveSettings : RefractiveIndex
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- StageLabel : Name
- StageLabel : Z
- TransmittanceRange : CutIn
- TransmittanceRange : CutOut

Total supported: 56

Total unknown or missing: 420

LeicaSCNReader

This page lists supported metadata fields for the Bio-Formats Leica SCN format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 33 of them (6%).
- Of those, Bio-Formats fully or partially converts 33 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Leica SCN format reader:

- Channel : ID
- Channel : IlluminationType
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : CalibratedMagnification
- Objective : ID
- Objective : LensNA
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY

- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 33

Total unknown or missing: 443

LiFlimReader

This page lists supported metadata fields for the Bio-Formats LI-FLIM format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 25 of them (5%).
- Of those, Bio-Formats fully or partially converts 25 (100%).

Supported fields

These fields are fully supported by the Bio-Formats LI-FLIM format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Image : ROIRef
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type

- Plane : DeltaT
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Polygon : ID
- Polygon : Points
- ROI : ID

Total supported: 25

Total unknown or missing: 451

MIASReader

This page lists supported metadata fields for the Bio-Formats MIAS format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 67 of them (14%).
- Of those, Bio-Formats fully or partially converts 67 (100%).

Supported fields

These fields are fully supported by the Bio-Formats MIAS format reader:

- Channel : Color
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Ellipse : ID
- Ellipse : RadiusX
- Ellipse : RadiusY
- Ellipse : Text
- Ellipse : TheT
- Ellipse : TheZ
- Ellipse : X
- Ellipse : Y
- Experiment : Description
- Experiment : ID
- Experiment : Type
- Image : AcquisitionDate

- Image : ExperimentRef
- Image : ID
- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- Instrument : ID
- Mask : BinData
- Mask : FillColor
- Mask : Height
- Mask : ID
- Mask : StrokeColor
- Mask : Width
- Mask : X
- Mask : Y
- Objective : ID
- Objective : Model
- Objective : NominalMagnification
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ColumnNamingConvention
- Plate : Columns

- Plate : ExternalIdentifier
- Plate : ID
- Plate : Name
- Plate : RowNamingConvention
- Plate : Rows
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : WellSampleRef
- ROI : ID
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index

Total supported: 67

Total unknown or missing: 409

MINCReader

This page lists supported metadata fields for the Bio-Formats MINC MRI format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 26 of them (5%).
- Of those, Bio-Formats fully or partially converts 26 (100%).

Supported fields

These fields are fully supported by the Bio-Formats MINC MRI format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID

- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 26

Total unknown or missing: 450

MNGReader

This page lists supported metadata fields for the Bio-Formats Multiple-image Network Graphics format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Multiple-image Network Graphics format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder

- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

MRCReader

This page lists supported metadata fields for the Bio-Formats Medical Research Council format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Medical Research Council format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ

- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

MRWReader

This page lists supported metadata fields for the Bio-Formats Minolta MRW format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Minolta MRW format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY

- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

MetamorphReader

This page lists supported metadata fields for the Bio-Formats Metamorph STK format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 59 of them (12%).
- Of those, Bio-Formats fully or partially converts 59 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Metamorph STK format reader:

- Channel : EmissionWavelength
- Channel : ID
- Channel : LightSourceSettingsID
- Channel : LightSourceSettingsWavelength
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : ReadOutRate
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- ImagingEnvironment : Temperature
- Instrument : ID

- Laser : ID
- Laser : LaserMedium
- Laser : Type
- Objective : ID
- Objective : LensNA
- ObjectiveSettings : ID
- ObjectiveSettings : RefractiveIndex
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ColumnNamingConvention
- Plate : Columns
- Plate : ID
- Plate : RowNamingConvention
- Plate : Rows
- Well : Column
- Well : ID

- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index

Total supported: 59

Total unknown or missing: 417

MetamorphTiffReader

This page lists supported metadata fields for the Bio-Formats Metamorph TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 43 of them (9%).
- Of those, Bio-Formats fully or partially converts 43 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Metamorph TIFF format reader:

- Channel : EmissionWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- ImagingEnvironment : Temperature
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX

- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ColumnNamingConvention
- Plate : Columns
- Plate : ID
- Plate : RowNamingConvention
- Plate : Rows
- PlateAcquisition : ID
- PlateAcquisition : WellSampleRef
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index

Total supported: 43

Total unknown or missing: 433

MetaxpressTiffReader

This page lists supported metadata fields for the Bio-Formats MetaXpress TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats MetaXpress TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

MicroCTReader

This page lists supported metadata fields for the Bio-Formats MicroCT format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 24 of them (5%).
- Of those, Bio-Formats fully or partially converts 24 (100%).

Supported fields

These fields are fully supported by the Bio-Formats MicroCT format reader:

- Channel : ID
- Channel : SamplesPerPixel

- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 24

Total unknown or missing: 452

MicromanagerReader

This page lists supported metadata fields for the Bio-Formats Micro-Manager format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 42 of them (8%).
- Of those, Bio-Formats fully or partially converts 42 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Micro-Manager format reader:

- Channel : Color
- Channel : ID

- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Manufacturer
- Detector : Model
- Detector : SerialNumber
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : Voltage
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- ImagingEnvironment : Temperature
- Instrument : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY

- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 42

Total unknown or missing: 434

MikroscanTiffReader

This page lists supported metadata fields for the Bio-Formats Mikroscan TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Mikroscan TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

MinimalTiffReader

This page lists supported metadata fields for the Bio-Formats Minimal TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Minimal TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

MolecularImagingReader

This page lists supported metadata fields for the Bio-Formats Molecular Imaging format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 21 of them (4%).
- Of those, Bio-Formats fully or partially converts 21 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Molecular Imaging format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 21

Total unknown or missing: 455

NAFReader

This page lists supported metadata fields for the Bio-Formats Hamamatsu Aquacosmos format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Hamamatsu Aquacosmos format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

ND2Reader

This page lists supported metadata fields for the Bio-Formats Nikon ND2 format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 52 of them (10%).
- Of those, Bio-Formats fully or partially converts 52 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Nikon ND2 format reader:

- Channel : AcquisitionMode
- Channel : Color
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : ReadOutRate
- DetectorSettings : Voltage
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- ImagingEnvironment : Temperature
- Instrument : ID
- Objective : CalibratedMagnification
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Model
- ObjectiveSettings : ID
- ObjectiveSettings : RefractiveIndex

- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 52

Total unknown or missing: 424

NDPIReader

This page lists supported metadata fields for the Bio-Formats Hamamatsu NDPI format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 33 of them (6%).
- Of those, Bio-Formats fully or partially converts 33 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Hamamatsu NDPI format reader:

- Channel : EmissionWavelength
- Channel : ID

- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Microscope : Model
- Objective : ID
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 33

Total unknown or missing: 443

NDPISReader

This page lists supported metadata fields for the Bio-Formats Hamamatsu NDPIS format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Hamamatsu NDPIS format reader:

- Channel : Color
- Channel : EmissionWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

NRRDReader

This page lists supported metadata fields for the Bio-Formats NRRD format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats NRRD format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

NiftiReader

This page lists supported metadata fields for the Bio-Formats NIfTI format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 24 of them (5%).
- Of those, Bio-Formats fully or partially converts 24 (100%).

Supported fields

These fields are fully supported by the Bio-Formats NIfTI format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 24

Total unknown or missing: 452

NikonElementsTiffReader

This page lists supported metadata fields for the Bio-Formats Nikon Elements TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 50 of them (10%).
- Of those, Bio-Formats fully or partially converts 50 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Nikon Elements TIFF format reader:

- Channel : AcquisitionMode
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : ReadOutRate
- DetectorSettings : Voltage
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- ImagingEnvironment : Temperature
- Instrument : ID
- Objective : CalibratedMagnification
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA

- Objective : Model
- ObjectiveSettings : ID
- ObjectiveSettings : RefractiveIndex
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 50

Total unknown or missing: 426

NikonReader

This page lists supported metadata fields for the Bio-Formats Nikon NEF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Nikon NEF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

NikonTiffReader

This page lists supported metadata fields for the Bio-Formats Nikon TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 47 of them (9%).
- Of those, Bio-Formats fully or partially converts 47 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Nikon TIFF format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength

- Channel : ID
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : Gain
- Detector : ID
- Detector : Type
- Dichroic : ID
- Dichroic : Model
- Filter : ID
- Filter : Model
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Laser : ID
- Laser : LaserMedium
- Laser : Model
- Laser : Type
- Laser : Wavelength
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : NominalMagnification
- Objective : WorkingDistance
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits

- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 47

Total unknown or missing: 429

OBFReader

This page lists supported metadata fields for the Bio-Formats OBF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats OBF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ

- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

OIRReader

This page lists supported metadata fields for the Bio-Formats Olympus OIR format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 48 of them (10%).
- Of those, Bio-Formats fully or partially converts 48 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Olympus OIR format reader:

- Channel : Color
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : LightSourceSettingsID
- Channel : Name
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : Gain
- Detector : ID
- Detector : Offset
- Detector : Voltage
- DetectorSettings : ID
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Laser : ID
- Laser : Model

- Laser : Wavelength
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Model
- Objective : NominalMagnification
- Objective : WorkingDistance
- ObjectiveSettings : ID
- ObjectiveSettings : RefractiveIndex
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 48

Total unknown or missing: 428

OMETiffReader

This page lists supported metadata fields for the Bio-Formats OME-TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats OME-TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

OMEXMLReader

This page lists supported metadata fields for the Bio-Formats OME-XML format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats OME-XML format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

OlympusTileReader

This page lists supported metadata fields for the Bio-Formats Olympus .omp2info format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Olympus .omp2info format reader:

- Channel : ID
- Channel : SamplesPerPixel

- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

OpenlabRawReader

This page lists supported metadata fields for the Bio-Formats Openlab RAW format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Openlab RAW format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder

- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

OpenlabReader

This page lists supported metadata fields for the Bio-Formats Openlab LIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 32 of them (6%).
- Of those, Bio-Formats fully or partially converts 32 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Openlab LIFF format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Type
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : Offset
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name

- Instrument : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 32

Total unknown or missing: 444

OperettaReader

This page lists supported metadata fields for the Bio-Formats PerkinElmer Operetta format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 60 of them (12%).
- Of those, Bio-Formats fully or partially converts 60 (100%).

Supported fields

These fields are fully supported by the Bio-Formats PerkinElmer Operetta format reader:

- Channel : AcquisitionMode
- Channel : Color
- Channel : ContrastMethod
- Channel : EmissionWavelength

- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Experimenter : ID
- Experimenter : LastName
- Image : AcquisitionDate
- Image : ExperimenterRef
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : ID
- Objective : LensNA
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC

- Plane : TheT
- Plane : TheZ
- Plate : Columns
- Plate : Description
- Plate : ExternalIdentifier
- Plate : ID
- Plate : Name
- Plate : Rows
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : StartTime
- PlateAcquisition : WellSampleRef
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index
- WellSample : PositionX
- WellSample : PositionY

Total supported: 60

Total unknown or missing: 416

OxfordInstrumentsReader

This page lists supported metadata fields for the Bio-Formats Oxford Instruments format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Oxford Instruments format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description

- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

PCIReader

This page lists supported metadata fields for the Bio-Formats Compix Simple-PCI format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 29 of them (6%).
- Of those, Bio-Formats fully or partially converts 29 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Compix Simple-PCI format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : ID

- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 29

Total unknown or missing: 447

PCORAWReader

This page lists supported metadata fields for the Bio-Formats PCO-RAW format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 26 of them (5%).
- Of those, Bio-Formats fully or partially converts 26 (100%).

Supported fields

These fields are fully supported by the Bio-Formats PCO-RAW format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- Detector : SerialNumber
- DetectorSettings : Binning
- DetectorSettings : ID
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Instrument : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 26

Total unknown or missing: 450

PCXReader

This page lists supported metadata fields for the Bio-Formats PCX format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats PCX format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

PDSReader

This page lists supported metadata fields for the Bio-Formats Perkin Elmer Densitometer format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 23 of them (4%).
- Of those, Bio-Formats fully or partially converts 23 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Perkin Elmer Densitometer format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 23

Total unknown or missing: 453

PGMReader

This page lists supported metadata fields for the Bio-Formats Portable Any Map format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Portable Any Map format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

PQBinReader

This page lists supported metadata fields for the Bio-Formats PicoQuant Bin format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 21 of them (4%).
- Of those, Bio-Formats fully or partially converts 21 (100%).

Supported fields

These fields are fully supported by the Bio-Formats PicoQuant Bin format reader:

- Channel : ID
- Channel : SamplesPerPixel

- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 21

Total unknown or missing: 455

PSDReader

This page lists supported metadata fields for the Bio-Formats Adobe Photoshop format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Adobe Photoshop format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name

- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

PerkinElmerReader

This page lists supported metadata fields for the Bio-Formats PerkinElmer format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 30 of them (6%).
- Of those, Bio-Formats fully or partially converts 30 (100%).

Supported fields

These fields are fully supported by the Bio-Formats PerkinElmer format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Pixels : BigEndian

- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 30

Total unknown or missing: 446

PhotoshopTiffReader

This page lists supported metadata fields for the Bio-Formats Adobe Photoshop TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Adobe Photoshop TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID

- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

PictReader

This page lists supported metadata fields for the Bio-Formats PICT format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats PICT format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved

- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

PovrayReader

This page lists supported metadata fields for the Bio-Formats POV-Ray format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats POV-Ray format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY

- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

PrairieReader

This page lists supported metadata fields for the Bio-Formats Prairie TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 46 of them (9%).
- Of those, Bio-Formats fully or partially converts 46 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Prairie TIFF format reader:

- Channel : EmissionWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Type
- Detector : Zoom
- DetectorSettings : Gain
- DetectorSettings : ID
- DetectorSettings : Offset
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Laser : ID
- Laser : Power
- Microscope : Model
- Objective : Correction

- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Manufacturer
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 46

Total unknown or missing: 430

PyramidTiffReader

This page lists supported metadata fields for the Bio-Formats Pyramid TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).

- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Pyramid TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

QTReader

This page lists supported metadata fields for the Bio-Formats QuickTime format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats QuickTime format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

QuesantReader

This page lists supported metadata fields for the Bio-Formats Quesant AFM format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Quesant AFM format reader:

- Channel : ID
- Channel : SamplesPerPixel

- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

RCPNLReader

This page lists supported metadata fields for the Bio-Formats RCPNL format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 26 of them (5%).
- Of those, Bio-Formats fully or partially converts 26 (100%).

Supported fields

These fields are fully supported by the Bio-Formats RCPNL format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID

- Image : Name
- Objective : Correction
- Objective : Immersion
- Objective : LensNA
- Objective : Manufacturer
- Objective : Model
- Objective : NominalMagnification
- Objective : WorkingDistance
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 26

Total unknown or missing: 450

RHKReader

This page lists supported metadata fields for the Bio-Formats RHK Technologies format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats RHK Technologies format reader:

- Channel : ID
- Channel : SamplesPerPixel

- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

SBIGReader

This page lists supported metadata fields for the Bio-Formats SBIG format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats SBIG format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description

- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

SDTReader

This page lists supported metadata fields for the Bio-Formats SPCImage Data format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats SPCImage Data format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian

- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

SEQReader

This page lists supported metadata fields for the Bio-Formats Image-Pro Sequence format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 20 of them (4%).
- Of those, Bio-Formats fully or partially converts 20 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Image-Pro Sequence format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits

- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 20

Total unknown or missing: 456

SIFReader

This page lists supported metadata fields for the Bio-Formats Andor SIF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 20 of them (4%).
- Of those, Bio-Formats fully or partially converts 20 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Andor SIF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ

- Pixels : Type
- Plane : DeltaT
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 20

Total unknown or missing: 456

SISReader

This page lists supported metadata fields for the Bio-Formats Olympus SIS TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 33 of them (6%).
- Of those, Bio-Formats fully or partially converts 33 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Olympus SIS TIFF format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- Detector : Type
- DetectorSettings : ID
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder

- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 33

Total unknown or missing: 443

SMCameraReader

This page lists supported metadata fields for the Bio-Formats SM Camera format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats SM Camera format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits

- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

SPCReader

This page lists supported metadata fields for the Bio-Formats SPC FIFO Data format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats SPC FIFO Data format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ

- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

SPEReader

This page lists supported metadata fields for the Bio-Formats Princeton Instruments SPE format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 30 of them (6%).
- Of those, Bio-Formats fully or partially converts 30 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Princeton Instruments SPE format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Image : ROIRef
- Label : ID
- Label : Text
- Label : X
- Label : Y
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ

- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- ROI : ID
- Rectangle : Height
- Rectangle : ID
- Rectangle : Width
- Rectangle : X
- Rectangle : Y

Total supported: 30

Total unknown or missing: 446

SVSReader

This page lists supported metadata fields for the Bio-Formats Aperio SVS format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 33 of them (6%).
- Of those, Bio-Formats fully or partially converts 33 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Aperio SVS format reader:

- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : ID
- Objective : NominalMagnification
- ObjectiveSettings : ID

- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 33

Total unknown or missing: 443

ScanrReader

This page lists supported metadata fields for the Bio-Formats Olympus ScanR format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 43 of them (9%).
- Of those, Bio-Formats fully or partially converts 43 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Olympus ScanR format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID

- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : ColumnNamingConvention
- Plate : Columns
- Plate : ID
- Plate : Name
- Plate : RowNamingConvention
- Plate : Rows
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : WellSampleRef
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index

- WellSample : PositionX
- WellSample : PositionY

Total supported: 43

Total unknown or missing: 433

SeikoReader

This page lists supported metadata fields for the Bio-Formats Seiko format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Seiko format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

SimplePCITiffReader

This page lists supported metadata fields for the Bio-Formats SimplePCI TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 33 of them (6%).
- Of those, Bio-Formats fully or partially converts 33 (100%).

Supported fields

These fields are fully supported by the Bio-Formats SimplePCI TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- Detector : Type
- DetectorSettings : Binning
- DetectorSettings : ID
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : ID
- Objective : Immersion
- Objective : NominalMagnification
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC

- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 33

Total unknown or missing: 443

SlideBook7Reader

This page lists supported metadata fields for the Bio-Formats SlideBook 7 SLD (native) format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 64 of them (13%).
- Of those, Bio-Formats fully or partially converts 64 (100%).

Supported fields

These fields are fully supported by the Bio-Formats SlideBook 7 SLD (native) format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Ellipse : ID
- Ellipse : RadiusX
- Ellipse : RadiusY
- Ellipse : X
- Ellipse : Y
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- Instrument : ID

- Line : ID
- Line : X1
- Line : X2
- Line : Y1
- Line : Y2
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : Model
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Point : ID
- Point : X
- Point : Y

- Polygon : ID
- Polygon : Points
- ROI : ID
- ROI : Name
- Rectangle : Height
- Rectangle : ID
- Rectangle : Text
- Rectangle : TheC
- Rectangle : TheT
- Rectangle : TheZ
- Rectangle : Width
- Rectangle : X
- Rectangle : Y

Total supported: 64

Total unknown or missing: 412

SlidebookReader

This page lists supported metadata fields for the Bio-Formats Olympus Slidebook format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 34 of them (7%).
- Of those, Bio-Formats fully or partially converts 34 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Olympus Slidebook format reader:

- Channel : ID
- Channel : NDFilter
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : Correction

- Objective : ID
- Objective : Immersion
- Objective : Model
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 34

Total unknown or missing: 442

SlidebookTiffReader

This page lists supported metadata fields for the Bio-Formats Slidebook TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 30 of them (6%).
- Of those, Bio-Formats fully or partially converts 30 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Slidebook TIFF format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Instrument : ID
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : NominalMagnification
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 30

Total unknown or missing: 446

SpiderReader

This page lists supported metadata fields for the Bio-Formats SPIDER format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 21 of them (4%).
- Of those, Bio-Formats fully or partially converts 21 (100%).

Supported fields

These fields are fully supported by the Bio-Formats SPIDER format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 21

Total unknown or missing: 455

TCSReader

This page lists supported metadata fields for the Bio-Formats Leica TCS TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 24 of them (5%).
- Of those, Bio-Formats fully or partially converts 24 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Leica TCS TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 24

Total unknown or missing: 452

TargaReader

This page lists supported metadata fields for the Bio-Formats Truevision Targa format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 20 of them (4%).
- Of those, Bio-Formats fully or partially converts 20 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Truevision Targa format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 20

Total unknown or missing: 456

TecanReader

This page lists supported metadata fields for the Bio-Formats Tecan Spark Cyto format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 36 of them (7%).
- Of those, Bio-Formats fully or partially converts 36 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Tecan Spark Cyto format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : Columns
- Plate : ID
- Plate : Name

- Plate : Rows
- PlateAcquisition : ID
- PlateAcquisition : MaximumFieldCount
- PlateAcquisition : WellSampleRef
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index

Total supported: 36

Total unknown or missing: 440

TextReader

This page lists supported metadata fields for the Bio-Formats Text format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Text format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY

- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

TiffDelegateReader

This page lists supported metadata fields for the Bio-Formats Tagged Image File Format format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Tagged Image File Format format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

TiffJAIReader

This page lists supported metadata fields for the Bio-Formats Tagged Image File Format format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Tagged Image File Format format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

TiffReader

This page lists supported metadata fields for the Bio-Formats Tagged Image File Format format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Tagged Image File Format format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

TileJPEGReader

This page lists supported metadata fields for the Bio-Formats Tile JPEG format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Tile JPEG format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

TillVisionReader

This page lists supported metadata fields for the Bio-Formats TillVision format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats TillVision format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Experiment : ID
- Experiment : Type
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

TissueFAXSReader

This page lists supported metadata fields for the Bio-Formats TissueFAXS format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 33 of them (6%).

- Of those, Bio-Formats fully or partially converts 33 (100%).

Supported fields

These fields are fully supported by the Bio-Formats TissueFAXS format reader:

- Channel : AcquisitionMode
- Channel : Color
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Instrument : ID
- Objective : ID
- Objective : Immersion
- Objective : LensNA
- Objective : Model
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT

- Plane : TheZ

Total supported: 33

Total unknown or missing: 443

TopometrixReader

This page lists supported metadata fields for the Bio-Formats TopoMetrix format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats TopoMetrix format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

TrestleReader

This page lists supported metadata fields for the Bio-Formats Trestle format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 27 of them (5%).
- Of those, Bio-Formats fully or partially converts 27 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Trestle format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Image : ROIRef
- Mask : BinData
- Mask : Height
- Mask : ID
- Mask : Width
- Mask : X
- Mask : Y
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC

- Plane : TheT
- Plane : TheZ
- ROI : ID

Total supported: 27

Total unknown or missing: 449

UBMReader

This page lists supported metadata fields for the Bio-Formats UBM format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats UBM format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

UnisokuReader

This page lists supported metadata fields for the Bio-Formats Unisoku STM format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Unisoku STM format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

VGSAMReader

This page lists supported metadata fields for the Bio-Formats VG SAM format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats VG SAM format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

VarianFDFReader

This page lists supported metadata fields for the Bio-Formats Varian FDF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 25 of them (5%).
- Of those, Bio-Formats fully or partially converts 25 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Varian FDF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 25

Total unknown or missing: 451

VectraReader

This page lists supported metadata fields for the Bio-Formats PerkinElmer Vectra/QPTIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 43 of them (9%).
- Of those, Bio-Formats fully or partially converts 43 (100%).

Supported fields

These fields are fully supported by the Bio-Formats PerkinElmer Vectra/QPTIFF format reader:

- Channel : Color
- Channel : Fluor
- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- Experimenter : ID
- Experimenter : UserName
- Image : AcquisitionDate
- Image : Description
- Image : ExperimenterRef
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Microscope : Model
- Objective : ID
- Objective : Model
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder

- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 43

Total unknown or missing: 433

VeecoReader

This page lists supported metadata fields for the Bio-Formats Veeco format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Veeco format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder

- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

VentanaReader

This page lists supported metadata fields for the Bio-Formats Ventana .bif format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 28 of them (5%).
- Of those, Bio-Formats fully or partially converts 28 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Ventana .bif format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : ID
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder

- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : PositionX
- Plane : PositionY
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 28

Total unknown or missing: 448

VisitechReader

This page lists supported metadata fields for the Bio-Formats Visitech XYS format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Visitech XYS format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID

- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

VolocityClippingReader

This page lists supported metadata fields for the Bio-Formats Volocity Library Clipping format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Volocity Library Clipping format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX

- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

VolocityReader

This page lists supported metadata fields for the Bio-Formats Volocity Library format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 38 of them (7%).
- Of those, Bio-Formats fully or partially converts 38 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Volocity Library format reader:

- Channel : ID
- Channel : Name
- Channel : SamplesPerPixel
- Detector : ID
- Detector : Model
- DetectorSettings : ID
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Instrument : ID
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian

- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 38

Total unknown or missing: 438

WATOPReader

This page lists supported metadata fields for the Bio-Formats WA Technology TOP format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 22 of them (4%).
- Of those, Bio-Formats fully or partially converts 22 (100%).

Supported fields

These fields are fully supported by the Bio-Formats WA Technology TOP format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : Description

- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 22

Total unknown or missing: 454

XLEFReader

This page lists supported metadata fields for the Bio-Formats Extended leica file format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Extended leica file format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian

- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

ZeissCZIReader

This page lists supported metadata fields for the Bio-Formats Zeiss CZI format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 181 of them (38%).
- Of those, Bio-Formats fully or partially converts 181 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Zeiss CZI format reader:

- Arc : ID
- Arc : LotNumber
- Arc : Manufacturer
- Arc : Model
- Arc : Power
- Arc : SerialNumber
- Channel : AcquisitionMode
- Channel : Color
- Channel : EmissionWavelength
- Channel : ExcitationWavelength
- Channel : FilterSetRef

- Channel : Fluor
- Channel : ID
- Channel : IlluminationType
- Channel : LightSourceSettingsAttenuation
- Channel : LightSourceSettingsID
- Channel : Name
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : AmplificationGain
- Detector : Gain
- Detector : ID
- Detector : LotNumber
- Detector : Manufacturer
- Detector : Model
- Detector : Offset
- Detector : SerialNumber
- Detector : Type
- Detector : Zoom
- DetectorSettings : Binning
- DetectorSettings : Gain
- DetectorSettings : ID
- Dichroic : ID
- Dichroic : LotNumber
- Dichroic : Manufacturer
- Dichroic : Model
- Dichroic : SerialNumber
- Ellipse : ID
- Ellipse : RadiusX
- Ellipse : RadiusY
- Ellipse : Text
- Ellipse : X
- Ellipse : Y
- Experimenter : Email
- Experimenter : FirstName
- Experimenter : ID
- Experimenter : Institution

- Experimenter : LastName
- Experimenter : MiddleName
- Experimenter : UserName
- Filament : ID
- Filament : LotNumber
- Filament : Manufacturer
- Filament : Model
- Filament : Power
- Filament : SerialNumber
- Filter : FilterWheel
- Filter : ID
- Filter : LotNumber
- Filter : Manufacturer
- Filter : Model
- Filter : SerialNumber
- Filter : Type
- FilterSet : DichroicRef
- FilterSet : EmissionFilterRef
- FilterSet : ExcitationFilterRef
- FilterSet : ID
- FilterSet : LotNumber
- FilterSet : Manufacturer
- FilterSet : Model
- FilterSet : SerialNumber
- Image : AcquisitionDate
- Image : Description
- Image : ExperimenterRef
- Image : ID
- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- ImagingEnvironment : AirPressure
- ImagingEnvironment : CO2Percent
- ImagingEnvironment : Humidity
- ImagingEnvironment : Temperature
- Instrument : ID

- Laser : ID
- Laser : LotNumber
- Laser : Manufacturer
- Laser : Model
- Laser : Power
- Laser : SerialNumber
- LightEmittingDiode : ID
- LightEmittingDiode : LotNumber
- LightEmittingDiode : Manufacturer
- LightEmittingDiode : Model
- LightEmittingDiode : Power
- LightEmittingDiode : SerialNumber
- Line : ID
- Line : Text
- Line : X1
- Line : X2
- Line : Y1
- Line : Y2
- Microscope : LotNumber
- Microscope : Manufacturer
- Microscope : Model
- Microscope : SerialNumber
- Microscope : Type
- Objective : CalibratedMagnification
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : Iris
- Objective : LensNA
- Objective : LotNumber
- Objective : Manufacturer
- Objective : Model
- Objective : NominalMagnification
- Objective : SerialNumber
- Objective : WorkingDistance
- ObjectiveSettings : CorrectionCollar

- ObjectiveSettings : ID
- ObjectiveSettings : Medium
- ObjectiveSettings : RefractiveIndex
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Plate : Columns
- Plate : ID
- Plate : Rows
- Point : ID
- Point : X
- Point : Y
- Polygon : ID
- Polygon : Points
- Polygon : Text
- Polyline : ID

- Polyline : Points
- Polyline : Text
- ROI : Description
- ROI : ID
- ROI : Name
- Rectangle : Height
- Rectangle : ID
- Rectangle : Text
- Rectangle : Width
- Rectangle : X
- Rectangle : Y
- StageLabel : Name
- StageLabel : X
- StageLabel : Y
- StageLabel : Z
- TransmittanceRange : CutIn
- TransmittanceRange : CutInTolerance
- TransmittanceRange : CutOut
- TransmittanceRange : CutOutTolerance
- TransmittanceRange : Transmittance
- Well : Column
- Well : ID
- Well : Row
- WellSample : ID
- WellSample : ImageRef
- WellSample : Index

Total supported: 181

Total unknown or missing: 295

ZeissLMSReader

This page lists supported metadata fields for the Bio-Formats Zeiss LMS format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 23 of them (4%).
- Of those, Bio-Formats fully or partially converts 23 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Zeiss LMS format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Instrument : ID
- Objective : ID
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 23

Total unknown or missing: 453

ZeissLSMReader

This page lists supported metadata fields for the Bio-Formats Zeiss Laser-Scanning Microscopy format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 101 of them (21%).
- Of those, Bio-Formats fully or partially converts 101 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Zeiss Laser-Scanning Microscopy format reader:

- Channel : Color
- Channel : ID
- Channel : Name
- Channel : PinholeSize
- Channel : SamplesPerPixel
- Detector : AmplificationGain
- Detector : Gain
- Detector : ID
- Detector : Type
- Detector : Zoom
- DetectorSettings : Binning
- DetectorSettings : ID
- Dichroic : ID
- Dichroic : Model
- Ellipse : FontSize
- Ellipse : ID
- Ellipse : RadiusX
- Ellipse : RadiusY
- Ellipse : StrokeWidth
- Ellipse : Transform
- Ellipse : X
- Ellipse : Y
- Experimenter : ID
- Experimenter : UserName
- Filter : ID
- Filter : Model
- Filter : Type
- Image : AcquisitionDate
- Image : Description
- Image : ID
- Image : InstrumentRef
- Image : Name
- Image : ROIRef
- Instrument : ID

- Label : FontSize
- Label : ID
- Label : StrokeWidth
- Label : Text
- Label : X
- Label : Y
- Laser : ID
- Laser : LaserMedium
- Laser : Model
- Laser : Type
- Laser : Wavelength
- LightPath : DichroicRef
- LightPath : EmissionFilterRef
- Line : FontSize
- Line : ID
- Line : StrokeWidth
- Line : X1
- Line : X2
- Line : Y1
- Line : Y2
- Objective : Correction
- Objective : ID
- Objective : Immersion
- Objective : Iris
- Objective : LensNA
- Objective : NominalMagnification
- ObjectiveSettings : ID
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC

- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : TimeIncrement
- Pixels : Type
- Plane : DeltaT
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ
- Polygon : FontSize
- Polygon : ID
- Polygon : Points
- Polygon : StrokeWidth
- Polyline : FontSize
- Polyline : ID
- Polyline : Points
- Polyline : StrokeWidth
- ROI : ID
- Rectangle : FontSize
- Rectangle : Height
- Rectangle : ID
- Rectangle : StrokeWidth
- Rectangle : Width
- Rectangle : X
- Rectangle : Y
- TransmittanceRange : CutIn
- TransmittanceRange : CutOut

Total supported: 101

Total unknown or missing: 375

ZeissTIFFReader

This page lists supported metadata fields for the Bio-Formats Zeiss AxioVision TIFF format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Zeiss AxioVision TIFF format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

ZeissXRMReader

This page lists supported metadata fields for the Bio-Formats Zeiss XRM format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 27 of them (5%).
- Of those, Bio-Formats fully or partially converts 27 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Zeiss XRM format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : PhysicalSizeX
- Pixels : PhysicalSizeY
- Pixels : PhysicalSizeZ
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : DeltaT
- Plane : ExposureTime
- Plane : PositionX
- Plane : PositionY
- Plane : PositionZ
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 27

Total unknown or missing: 449

ZeissZVIReader

This page lists supported metadata fields for the Bio-Formats Zeiss Vision Image (ZVI) format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Zeiss Vision Image (ZVI) format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

ZipReader

This page lists supported metadata fields for the Bio-Formats Zip format reader.

These fields are from the OME data model. Bio-Formats standardizes each format's original metadata to and from the OME data model so that you can work with a particular piece of metadata (e.g. physical width of the image in microns) in a format-independent way.

Of the 476 fields documented in the *metadata summary table*:

- The file format itself supports 19 of them (3%).
- Of those, Bio-Formats fully or partially converts 19 (100%).

Supported fields

These fields are fully supported by the Bio-Formats Zip format reader:

- Channel : ID
- Channel : SamplesPerPixel
- Image : AcquisitionDate
- Image : ID
- Image : Name
- Pixels : BigEndian
- Pixels : DimensionOrder
- Pixels : ID
- Pixels : Interleaved
- Pixels : SignificantBits
- Pixels : SizeC
- Pixels : SizeT
- Pixels : SizeX
- Pixels : SizeY
- Pixels : SizeZ
- Pixels : Type
- Plane : TheC
- Plane : TheT
- Plane : TheZ

Total supported: 19

Total unknown or missing: 457

4.4 Grouping files using a pattern file

Individual files can be grouped together into a single fileset using a pattern file. This works for any single-file format that Bio-Formats supports, as long as all files are in the same format. It is most useful for sets of TIFF, JPEG, PNG, etc. files that do not have any associated metadata.

All files to be grouped together should be in the same folder. The pattern file should be in the same folder as the other files; it can have any name, but must have the `.pattern` extension. The pattern file is what must be opened or imported, so it may be helpful to give it a descriptive or easily-recognizable name.

The pattern file contains a single line of text that is specially formatted to describe how the files should be grouped. The file can be created in any text editor.

The text in the pattern file can take one of several forms. To illustrate, consider a folder with the following file names:

```
red.tiff
green.tiff
blue.tiff
test_Z0_C0.png
test_Z1_C0.png
test_Z0_C1.png
test_Z1_C1.png
test_Z0_C2.png
test_Z1_C2.png
test_Z00.tiff
test_Z01.tiff
```

A pattern file that groups `red.tiff`, `green.tiff`, and `blue.tiff` in that order would look like:

```
<red,green,blue>.tiff
```

A pattern that groups `test_Z0_C0.png`, `test_Z1_C0.png`, `test_Z0_C2.png`, and `test_Z1_C2.png`:

```
test_Z<0-1>_C<0-2:2>.png
```

The <> notation in general can accept a single literal value, a comma-separated list of literal values, a range of integer values, or a range of integer values with a step value greater than 1 (the range and step are separated by `:`). Note that inverting the values in a range (e.g. `<2-0>`) is not supported and will cause an exception to be thrown.

The characters immediately preceding the < can affect which dimension is assigned to the specified values. The values will be interpreted as:

- channels, if `c`, `ch`, `w`, or `wavelength` precede <
- timepoints, if `t`, `tl`, `tp`, or `timepoint` precede <
- Z sections, if `z`, `zs`, `sec`, `fp`, `focal`, or `focalplane` precede <
- series, if `s`, `sp`, or `series` precede <

Note that the listed dimension specifier characters are case insensitive. A separator character (underscore or space) must precede the dimension specifier if it is not at the beginning of the filename. In the above example, 2 Z sections and 2 out of 3 channels would be detected according to the dimension specifiers.

Leading zeros in the integer values must be specified. To group `test_Z00.tiff` and `test_Z01.tiff`:

```
test_Z<00-01>.tiff
```

or:

```
test_Z0<0-1>.tiff
```

Note that this pattern would not group the files correctly:

```
test_Z<0-1>.tiff
```

A pattern file that groups all PNG files beginning with `test_` would look like:

```
test_.*.png
```

This and most other Java-style regular expressions can be used in place of the <> notation above. See the java.util.regex.Pattern Javadoc for more information on constructing regular expressions.

4.5 Support for “precompressed” tiles

Some formats allow reading and/or writing “precompressed” tiles. This feature is primarily intended for whole slide images, each of which may have thousands of tiles. Data representing these tiles may be compressed, either lossily or losslessly, depending on the file format, imaging modality, or acquisition configuration.

When reading, this allows for retrieving compressed bytes that are as close as possible to what is actually stored in the file. Decompression (if necessary) is the responsibility of the consumer.

When writing, compressed bytes instead of raw bytes can be provided to the writer, such that the writer will just save the provided compressed bytes instead of recompressing.

See also *how to work with precompressed tiles in bfconvert*

4.5.1 Formats that support precompressed tile reading

- SVS (since 7.1.0)
- NDPI (since 8.0.0)
- DICOM (since 8.0.0)

4.5.2 Formats that support precompressed tile writing

- DICOM (since 7.1.0)
- TIFF (since 8.0.0)
- OME-TIFF (since 8.0.0)

4.6 Additional reader and writer options

Some readers and writers have additional options which can be used to inform how Bio-Formats reads or writes files in that format.

4.6.1 Reader options

Format name	Option	Default	Description
cellSens VSI	`cellsens.fail_on_miss`	false	Throw an exception if an expected associated .ets file is missing
Gatan Digital Micrograph	`gatan.split_montag`	true	Split montage image tiles across multiple planes
Leica LAS AF LIF (Leica Image File Format)	`leicalif.old_physical`	false	Ensure physical pixel sizes are compatible with versions <= 5.3.2
Nikon NIS-Elements ND2	`nd2.chunkmap`	true	Use chunkmap table to read image offsets
Olympus ScanR	`scanr.skip_missing`	true	Ignore missing wells
OME-TIFF	`ometiff.fail_on_miss`	true	Fail if a missing file is detected in a partial dataset
Ventana BIF	`ventana.split_tiles`	false	Report each tile as a separate series
Zeiss CZI	`zeissczi.attachments`	true	Include attachment images
Zeiss CZI	`zeissczi.autostitch`	true	Automatically stitch tiled images
Zeiss CZI	`zeissczi.trim_dimensi`	false	Trim XY dimensions to match those in ZEN
Zeiss CZI	`zeissczi.relative_pos`	false	Use pixel position instead of the physical stage position
CV7000	`cv7000.duplicate_mi`	true	If set any missing planes will be replaced by the first plane in the channel. When set to false a blank plane will be returned.
Aperio SVS TIFF	`svs.remove_thumb`	true	If set the thumbnail image (if present) will not be available, and in particular will not be included in the pyramid. When set to false, the thumbnail image will be available for compatibility with versions <= 7.1.0.
InCell 1000/2000	`incell.duplicate_mi`	true	If set any missing planes will be replaced by the first plane in the channel. When set to false a blank plane will be returned.

Usage

Reader options can be used via the command line with `showinf -option`, in ImageJ via the *configuration window*, or via the API using the DynamicMetadataOptions class.

4.6.2 Writer options

Format name	Option	Default	Description
OME-TIFF	`ometif compan`	None	If set, OME-XML will be written to a companion file with a name determined by the option value
OME-TIFF	`ometif preser`	false	If set to false, the OME-XML Creator attribute will be updated to include the current Bio-Formats version. This will overwrite any existing Creator in the input metadata. If set to true, the Creator attribute will only be updated if it wasn't previously set; any existing value will be preserved.
OME-XML	`omexml preser`	false	If set to false, the OME-XML Creator attribute will be updated to include the current Bio-Formats version. This will overwrite any existing Creator in the input metadata. If set to true, the Creator attribute will only be updated if it wasn't previously set; any existing value will be preserved.
DICOM	`dicom. dual_p`	true	If set to true, DICOM files will be written as dual-personality DICOM/TIFF files which include TIFF metadata as described in Clunie DA. Dual-Personality DICOM-TIFF for Whole Slide Images: A Migration Technique for Legacy Software. J Pathol Inform. 2019 Apr 3;10:12. doi: 10.4103/jpi.jpi_93_18. PMID: 31057981; PMCID: PMC6489422.. If set to false, the writing of DICOM files will revert back to the original plain DICOM files

Usage

Writer options can be used via the command line using *`bfconvert -option`*, or via the API using the DynamicMetadataOptions class.

CHAPTER

FIVE

VERSION HISTORY

5.1 8.5.0 (2026 March)

File format fixes:

- **Amira**
  - improve physical size unit detection
- **DICOM**
  - check *Study Time* tag when determining which files belong in the same dataset
  - enforce correct value size when writing *DS* VR
- **Evident/Olympus VSI**
  - recognize calibration function tag and store as *Calibration Function Value* in original metadata
- **Hamamatsu NDPIS**
  - preserve macro/mask image names, consistent with underlying NDPI data
- **Imaris HDF**
  - work around issue in netcdf-java which prevented files with # in path from being opened
- **TIFF**
  - improve file size estimate when determining whether to write BigTIFF
- **Zeiss CZI**
  - ignore plate metadata if a plate configuration is present but not activated
  - fix calculation of phase, illumination, and rotation counts

Bio-Formats improvements:

- Fix many Javadoc warnings
- Unify XML parsing using changes in *ome-common* 6.2.1 to fix CVE-2026-22186, see corresponding security advisory

Component updates:

- *ome-codecs* was upgraded to 1.1.3
- *ome-common* was upgraded to 6.2.1
- *ome-model* was upgraded to 6.5.3
- *formats-gpl* now inherits *aircompressor* dependency from *ome-codecs*

- Remove outdated dependency exclusions from *formats-gpl*

5.2 8.4.0 (2026 January)

New file formats:

- Added support for TissueGnostics TFCYTO file format, sponsored by TissueGnostics GmbH

File format fixes:

- **CellH5**
  - reduce memory usage
- **DCIMG**
  - fix indexing over multiple files (thanks to Zach Marin)
- **Evident/Olympus OIR**
  - fix several issues that prevented pixel blocks from being found
- **Evident/Olympus VSI**
  - parse timestamps for each plane if available
- **FV1000**
  - fix precision of ROI coordinate calculation
- **Hamamatsu NDPI**
  - read emission wavelength from single NDPI files
- **Leica TCS**
  - remove unused detector parsing logic
- **LEO**
  - fix tag parsing to use *iso-8859-1* encoding instead of *UTF-8*
- **Metamorph**
  - fix plane position parsing for multi-series data
- **Nikon ND2**
  - fix dimensions and timestamp assignment for certain multi-channel datasets
  - ignore invalid bits per pixel values when determining pixel type
  - prevent pixel types other than *uint8* from being set incorrectly
  - changed the plane padding for another condition to pad to 4 byte boundaries (thanks to Matthew Smith)
- **OME-XML**
  - fix *OriginalMetadataAnnotation* parsing to allow escaped characters
- **RCPNL**
  - define objective metadata for lens IDs 18113 and 18114

Bio-Formats improvements:

- **bfconvert**

  - automatically calculate a pyramid resolution count if *-pyramid-scale* is used without *-pyramid-resolutions*
  - prevent large values of *-pyramid-resolutions* from producing resolutions with invalid dimensions
  - fix pyramid generation when writing a subset of planes (with *-channel*, *-z*, and/or *-t*)
- **ImageJ plugin**
  - fix modulo C (e.g. lifetime) scrollbars
- **MATLAB/Octave toolbox**
  - fix array sizes in *bfGetPlane* when using Octave
- ***FakeReader***
  - add *labelPlanes* attribute to include human-readable plane indexes in each image
  - link *Instrument* to every *Image*
- Add unit tests for *ChannelMerger* and *ChannelSeparator*
- Several updates to automated repository tests to eliminate invalid test failures
- Automatically generate release notes when creating a GitHub release

Component updates:

- *logback* was upgraded to 1.3.16
- *ome-common* was upgraded to 6.1.2
- *ome-codecs* was upgraded to 1.1.2
- *ome-metakit* was upgraded to 5.3.10
- *ome-model* was upgraded to 6.5.1
- *ome-poi* was upgraded to 5.3.11
- CodeQL updated to v4

5.3 8.3.0 (2025 August)

New file formats:

- **Zeiss XRM:**
  - added support for .txm and .txrm files

File format fixes:

- **LiFlim**
  - cast integers to double for division accuracy
- **Metamorph**
  - don't reset dimensions if the well count and series count match
  - fixed the encoding used for reading the .nd file
- **Metamorph TIFF**
  - threw more explicit exceptions if parsing invalid TIFF
- **Nikon ND2**

  - added TextInfoItems to original metadata before parsing
  - changed the padding for a specific condition to pad to 4 byte boundaries (thanks to Matthew Smith)
  - removed the usage of position metadata when distinguishing between T and series
- **OME-TIFF**
  - fixed optimal tile size if an IFD is not linked to an Image
- **Operetta/Harmony**
  - fixed case where index XML is in the parent of the image directory
- **SlideBook 7**
  - fixed miscellaneous errors (thanks to Nicola Papp)
- **Tecan Spark Cyto Workspace**
  - used try-with-resources when opening SQL connection
- **Ventana BIF**
  - fixed LEFT overlap and added Z stack support
- **Zeiss CZI**
  - fixed LightSource population to include IDs, link LightSources to Channels, and set Attenuation
  - fixed IDs and reference issues for Dichroic and Detector objects
  - used millisecond units for exposure times attached to channels

Bio-Formats improvements:

- showinf: configure logging for *-omexml-only* after all arguments parsed
- xmlvalid: used cached XSD schemas for offline validation
- ChannelMerger: correct number of Channels in MetadataStore
- Check modulo values in repository tests
- Added workflow to report the status of the automated repository builds

Component updates:

- *commons-lang* was migrated to *commons-lang3* and upgraded to 3.18.0
- *metakit* was upgraded to 5.3.9
- *ome-common* was upgraded to 6.1.0
- *ome-codecs* was upgraded to 1.1.1
- *ome-jai* was upgraded to 0.1.5
- *ome-mdbtools* was upgraded to 5.3.4
- *ome-metakit* was upgraded to 5.3.9
- *ome-model* was upgraded to 6.5.0
- *ome-poi* was upgraded to 5.3.10
- *ome-stubs* was upgraded to 6.0.3

5.4 8.2.0 (2025 May)

New file formats:

- **Molecular Devices JDCE**
  - Added support for datasets acquired on ImageXpress HCS.ai High-Content Screening System
  - Added support for datasets acquired on IN Cell Analyzer 2XXX/6XXXX series and converted into JDCE format by IN Carta® Image Analysis Software

File format fixes:

- **ARF**
  - improve type checking to prevent false positives
- **Hamamatsu DCIMG**
  - fix image stride in DCIMG version 1 and later (thanks to Zach Marin)
- **Leica LIF**
  - fix mapping of images to memory blocks containing pixel data
- **Metamorph**
  - warn if timestamp values are invalid or cannot be parsed
- **MIAS**
  - fix sorting of directory listings
- **Prairie Technologies TIFF**
  - add support for multi-page TIFFs (thanks to Michael Fox)
- **ScanR**
  - make sure the initialized file is placed first on the used file list
- **Zeiss CZI**
  - fix sub-block sizes in pyramid sub-resolutions (thanks to Jürgen Bohl)

Bio-Formats improvements:

- Fix Maven repository URLs to prevent build instability
- Prevent file appending when exporting via ImageJ plugin
- Fix color map array allocation in *bfopen* function in MATLAB toolbox
- Remove obsolete Python script for updating version metadata
- Fix several type conversion warnings raised by CodeQL, to prevent integer overflows

Component updates:

- *sqlite-jdbc* was upgraded to 3.49.1.0 to fix CVE-2023-32697 (thanks to Curtis Rueden)

5.5 8.1.1 (2025 March)

File format fixes:

- **Imspector**
  - fix *NumberFormatException* when reading timepoint count

- **InCell**
  - ignore timepoint configurations if a timelapse was not acquired
  - fix handling of channels with different Z sizes
  - add *incell.duplicate_missing_planes* option
- **Metamorph**
  - allow files to initialize when channel wavelengths are invalid
- **Nikon ND2**
  - fix pixel type detection for floating point multiposition data
- **ScanR**
  - fix exposure time reporting when field positions are missing
- **Spider**
  - fix type cast warning in image count calculation raised by CodeQL

Bio-Formats improvements:

- Remove unused and obsolete C++ and Javascript files
- Update GitHub Actions configuration to include CodeQL workflows
- Add options to *FakeReader* to create test data with instrument and channel wavelength metadata
- Ensure *getAvailableOptions()* in *IFormatReader* API returns the correct reader options

5.6 8.1.0 (2025 February)

New file formats:

- **Hamamatsu DCIMG**
  - add a new reader for Hamamatsu .dcimg files (thanks to Zach Marin)

File format fixes:

- **Evident/Olympus OIR**
  - fix an issue with reading timestamp metadata
- **Evident/Olympus VSI**
  - fix ordering of used files list
- **Fuji**
  - fix ordering of used files list
- **Imaris HDF**
  - add caching and support for LZ4 compression (thanks to Marco Stucchi)
- **Leica TCS**
  - ensure all relevant files are included in used files list

Bio-Formats improvements:

- Remove *LOCI* entry from ImageJ plugins menu (thanks to Tiago Ferreira)
- Fix array endianness in *bfsave* MATLAB function

- Fix *-version*, *-debug*, *-trace*, and *-no-upgrade* options in *tiffcomment* command line tool

Component updates:

- *logback* was upgraded to 1.3.15
- *ome-codecs* was upgraded to 1.1.0
- *ome-common* was upgraded to 6.0.25
- *ome-metakit* was upgraded to 5.3.8
- *ome-model* was upgraded to 6.4.0

5.7 8.0.1 (2024 November)

File format fixes:

- **PerkinElmer Operetta**
  - prevent directories from being included in the used files list
  - calculate physical Z size in micrometers, for improved readability

Bio-Formats improvements:

- Fix regression in *showinf* that prevented images from being displayed
- Add test to ensure that the initialized file is placed first on the used files list
- Include original metadata when exporting OME-TIFF via the Bio-Formats Exporter plugin for ImageJ

5.8 8.0.0 (2024 October)

File format fixes and improvements:

- **CV7000**
  - allow TIFF files to be in a subdirectory
- **DICOM**
  - add support for reading precompressed tiles[1]
  - improve image write performance[1]
  - fix writing to DICOM when physical size units are missing from input data
- **Evident/Olympus OIR**
  - improve image read performance (thanks to Nicolas Chiaruttini)
- **Hamamatsu NDPI**
  - add support for reading precompressed tiles[1]
- **Leica LIF**
  - fix file offset calculations for datasets with unstitched tiles
- **OME-TIFF**
  - improve image write performance

[1] This functionality was implemented through collaboration between Glencoe Software and NCI Imaging Data Commons, and has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, under Task Order No. HHSN26110071 under Contract No. HHSN2612015000031.

  - add support for writing precompressed tiles[Page 457, 1]

- **OME-XML**
  - fix incorrect logical operators when checking image dimensions
- **RCPNL**
  - define objective metadata for lens ID 18112
- **SDT**
  - fix reading of files acquired with SPCM 9.90 (thanks to Chen Qian)
- **Zeiss LSM**
  - fix incorrect logical operators when setting channel colors

Bio-Formats improvements:

- Add a batch processing feature to *showinf* (thanks to Torsten Stöter)
- Update *bfconvert* to automatically detect the compression type when using *-precompressed*[Page 457, 1]
- Fix autoscaling of *float* values in *AWTImageTools*
- ImageJ export plugin: fix handling of multi-channel datasets
- ***JPEGTurboService***[Page 457, 1]
  - add several new methods for retrieving tiles from a single JPEG stream
  - expand Javadocs for existing methods

Documentation updates:

- **update all links to AwareSystems TIFF documentation to last version from Internet Archive**
  - see libtiff mailing list discussion for context
- fix index of command-line tools
- add documentation for new batch processing feature in *showinf* (thanks to Torsten Stöter)
- add documentation for precompressed tiles feature[Page 457, 1]
- several other fixes for broken links

Component updates:

- *ome-common* was upgraded to 6.0.24
- *cdm-core* was upgraded to 5.6.0

5.9 7.3.1 (2024 July)

File format fixes and improvements:

- **Aperio SVS**
  - updated isThisType to reject files with a single IFD
- **ICS (Image Cytometry Standard)**
  - fixed a Null Pointer Exception when the image name is not set
- **Imaris HDF**
  - fixed handling of channels with an empty color attribute

- **PerkinElmer Columbus**
  - updated to use micrometers instead of reference frame for positions

Bio-Formats improvements:

- **fixed warnings across all components for Java 9+**
  - Among other things, this includes changes to how double values are parsed from strings in many format readers. These changes are expected to be more lenient in allowing datasets with certain missing or invalid metadata values (e.g. objective magnifications, detector voltages) to be read without throwing an exception. This means that some datasets which failed to read in 7.3.0 may now suddenly work (or fail at a later point) with 7.3.1, without an intentional targetted fix for that format. For specific changes to format readers, please see https://github.com/ome/bioformats/pull/4182 and https://github.com/ome/bioformats/pull/4177.
- fixed a number of String comparisons

Documentation improvements:

- fixed a number of broken external links

5.10 7.3.0 (2024 April)

File format fixes and improvements:

- **CellSens VSI**
  - fixed an exception in CellSens 4.1 data by updating the handling of orphaned ets files (thanks to Edward Scanlon)
- **DICOM**
  - updated handling of nested sequences that include pixel data
  - removed a duplicate enum check in the DICOM writer
- **OME-XML**
  - reading an OME-XML file now populates the original metadata table
- **PerkinElmer Columbus**
  - added support for reading data with Z stacks (thanks to Tim Blackmore)
- **Prairie Technologies TIFF**
  - improved type checking for Prairie files
- **SimplePCI**
  - fixed handling of Z position metadata

Bio-Formats improvements:

- Bio-Formats is now compatible with Java 21 builds
- added a new *quality* option to *bfconvert* which enables the setting of the compression quality to be used.
- when using the *channel* option in *bfconvert*, the unselected channels are now correctly removed from the OME-XML metadata
- added coverage of Apple M1 as part of the CI process
- added coverage of Java 21 builds as part of the CI process

Component updates:

- *ome-jai* was upgraded to 0.1.4
- *ome-stubs* was upgraded to 6.0.2
- *ome-mdbtools* was upgraded to 5.3.3
- *ome-common* was upgraded to 6.0.22
- *ome-poi* was upgraded to 5.3.9
- *ome-metakit* was upgraded to 5.3.7
- *ome-codecs* was upgraded to 1.0.3
- *ome-xml* was upgraded to 6.3.6
- *specification* was upgraded to 6.3.6

Documentation improvements:

- fixed a number of broken external links
- documented the new *quality* option in *bfconvert*

5.11 7.2.0 (2024 February)

File format fixes and improvements:

- **Aperio SVS**
  - improved detection of label and macro images
  - thumbnails are now removed by default
  - added a new reader option *svs.remove_thumbnail* to control the thumbnail behaviour. By default the option is true and thumbnails are removed. Setting the option to false will restore the previous behavior and include thumbnails
- **Imaris HDF**
  - channel names are now correctly populated
- **Olympus OIR**
  - fixed an issue with channel ordering
  - prevent certain pixel blocks from being incorrectly skipped
- **TIFF**
  - dimension order is now marked as certain if metadata is read from ImageJ tag
- **Zeiss CZI**
  - improved well column indexing by using alternate sources for well identifiers when available
  - improved image naming, particularly for attachments
  - fixed a bug so that opening CZI compressed with ZSTD now allows access to pyramid levels (thanks to Stephan Wagner-Conrad)
  - fixed an integer arithmetic problem when calculating pyramid scales (thanks to Stephan Wagner-Conrad)

Bio-Formats improvements:

- JPEG handling no longer fails for non 4:4:4 JPEG (thanks to Can Gokhan Dogan)

- fixed a bug in the Bio-Formats Plugins configuration window when toggling the Nikon ND2 chunkmap option
- updated testing to add an additional check for image names when resolutions are not flattened

Documentation improvements:

- fixed a number of broken external links

5.12 7.1.0 (2023 December)

API Additions

- **ICompressedTileReader**
  - a new interface allowing for the reading of compressed bytes without decompression. Currently only implemented by the SVSReader
- **ICompressedTileWriter**
  - a new interface allowing for the writing of precompressed bytes. Currently only implemented by the DICOM Writer
- **IExtraMetadataWriter**
  - a new interface enabling extra supplemental metadata to be set for inclusion in a writer. Currently only implemented by the DICOM Writer

File format fixes and improvements:

- **Aperio SVS**
  - added implementation for the reading of compressed data without decompression using the new ICompressedTileReader interface
- **cellSens VSI**
  - fixed handling of uncompressed BGR data
- **CV7000**
  - action index is now included in the channel name
  - Fluor will now be set on Channel when possible
  - for missing planes the default behavior has switched from returning a blank plane to returning the first plane in the channel
  - added a new reader option *cv7000.duplicate_missing_planes* to control the behavior for missing planes
- **DICOM**
  - added implementation for the writing of precompressed data using the new ICompressedTileWriter interface
  - added implementation for the writing of extra supplemental metadata data using the new IExtraMetadataWriter interface. Extra metadata can be provided in the form of a .json file
  - This functionality was implemented through collaboration between Glencoe Software and NCI Imaging Data Commons, and has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, under Task Order No. HHSN26110071 under Contract No. HHSN2612015000031.
- **Photoshop TIFF**
  - fixed plane reading with memoization

Bio-Formats improvements:

- fixed handling of JPEG Turbo service when the native library cannot be loaded, most notably for Apple Silicon instances. An IOException will now be thrown by readers that require the library but the RunTimeException introduced in 7.0.1 will no longer be thrown when the library is not required
- prevented unnecessary Lurawave debug error being printed (thanks to Can Gokhan Dogan)
- updated license headers for all Java sources of the OME-XML library
- added a new codec option *disableChromaSubsampling* along with implementation for the JPEG Codec (thanks to Peter Haub)
- updated github actions checkout version to v4 and build release on ubuntu 22.04
- json dependency moved from formats-gpl to formats-bsd
- TiffSaver now generates valid 4-channel RGBA images (thanks to Can Gokhan Dogan)
- added a new ` precompressed` option to the bfconvert tool, enabling the conversion of precompressed tiles. Currently only supported for converting from SVS to DICOM.
- added a new ` extra-metadata` option to the bfconvert tool for providing supplemental metadata during conversion. Currently only supported for writing to DICOM

Component updates:

- *slf4j-api* was upgraded to 2.0.9
- *logback-classic* was upgraded to 1.3.14
- *logback-core* was upgraded to 1.3.14
- *xalan* was upgraded to 2.7.3
- *com.google.guava* was upgraded to 32.0.0-jre
- *json* was upgraded to 20230227
- *ome-common* was upgraded to 6.0.21
- *ome-model* was upgraded to 6.3.4
- *ome-codecs* was upgraded to 1.0.1
- *ome-poi* was upgraded to 5.3.8
- *ome-metakit* was upgraded to 5.3.6

Documentation improvements:

- fixed a number of broken external links
- updated the bfconvert instructions to document new options
- added a new link for the external Bioformats.NET project

5.13 7.0.1 (2023 October)

File format fixes and improvements:

- **Amersham Biosciences Gel**
  - improved format detection
- **Aperio SVS**
  - fixed a NullPointerException when attempting to read the magnification metadata

  - physical sizes and positions are now calculated for full resolution label and macro images
  - fixed a regression from Bio-Formats 6.8.0 which saw label and macro images incorrectly indexed
- **DICOM**
  - the DICOM writer will now throw an exception if the provided tiles don't match the expected tile size
- **Gatan DM3**
  - fixed a StringIndexOutOfBoundsException when parsing acquisition mode
- **Hamamatsu NDPI**
  - corrected for potential integer overflow in stored restart marker tag
- **IVision**
  - improved the version and data type checks in isThisType
- **Leica LIF**
  - improved the exception handling if no images are found
- **Olympus OIR**
  - fixed issue with XML block reading that resulted in missing pixel blocks
- **OME-TIFF**
  - reader will now fail fast and throw a FormatException if an invalid file is found
- **TIFF**
  - a FormatException is now thrown for unsupported 64-bit int data
  - for all TIFF based formats the IFD entry offset is now checked before seeking
- **TillVision**
  - improved metadata parsing and detection of .pst files
- **Ventana**
  - fixed tile arithmetic for smallest resolutions
- **Zeiss CZI**
  - unified channel color parsing so that retrieving lookup tables now allows RGB and ARGB colors

Bio-Formats improvements:

- URLHandle will fall back to resetting stream if resetting to a mark fails
- an exception will now be thrown if the TurboJPEG library fails to load (thanks to Can Gokhan Dogan)
- fixed a Bio-Formats plugin bug when attempting to export a file after reducing the range of channels
- reviewed and updated a number of file headers
- updated GitHub Actions from checkout v2 to v3

Component updates:

- *ome-common* was upgraded to 6.0.20

Documentation improvements:

- fixed a number of broken external links
- updated the link to OME Model documentation

- migrated readthedocs configuration to version 2

5.14 7.0.0 (2023 August)

Removal of deprecated components:

- **Legacy ND2 Reader**
  - The LegacyND2Reader and underlying components have been removed. This reader depended on an outdated DLL which had not been built in years, was untested and fully superseded by the new NativeND2Reader. The existing NativeND2Reader has been renamed as ND2Reader and now acts as the sole reader for the format. Support for the ND2 format will be unaffected and continue via this newly renamed reader.
- **Legacy QuickTime**
  - The LegacyQT Reader, Writer and Tools have been removed. These classes depended on the external QuickTime for Java library which is long obsolete. The existing NativeQTReader has been renamed as QTReader and support for QuickTime is now limited to non legacy readers and writers.
- **LuraWave Codec**
  - The LuraWave Codec along with the LuraWaveService and associated components have been removed and are no longer supported. This codec was dependent on a third-party proprietary library with a license code. This may impact LuraWave compressed files for the Opera Flex format.
- **Woolz**
  - The Woolz Reader, Writer and Service have been removed and are no longer supported. The reader and writer are untested and rely on an underlying library which has an unclear maintenance status.

File format fixes and improvements:

- **DICOM**
  - support has been added for the writing dual-personality of DICOM/TIFF. The DICOM writer will now, by default, include TIFF metadata as described in https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6489422/. These new dual-personality files can be read either by the DICOM reader or can also be recognised as TIFF. A new option named *dicom.dual_personality* has been added to the DicomWriter. Setting this new option to false will revert writing of DICOM files back to the original plain DICOM files. This functionality was implemented through collaboration with NCI Imaging Data Commons, and has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, under Task Order No. HHSN26110071 under Contract No. HHSN2612015000031.
- **PerkinElmer Opera Flex**
  - reader will now ignore files in which the compression cannot be identified
- **PerkinElmer Operetta**
  - fixed an exception when reading null planes (thanks to Nicolas Chiaruttini)

Component updates:

- *ome-common* was upgraded to 6.0.19
- *ome-stubs* was upgraded to 6.0.1
- *ome-codecs* was upgraded to 1.0.0
- *native-lib-loader* was upgraded to 2.4.0 (thanks to Can Gokhan Dogan)

Documentation improvements:

- fixed a number of broken links

5.15 6.14.0 (2023 July)

Deprecation warnings:

- **Legacy ND2 Reader**
  - The LegacyND2Reader and underlying components have been marked as deprecated in preparation for removal in the upcoming 7.0.0 major release of Bio-Formats. This reader depends on an outdated DLL which has not been built in years, is untested and fully superseded by the new NativeND2Reader. From Bio-Formats 7.0.0 onwards the existing NativeND2Reader will be renamed as ND2Reader and act as the sole reader for the format. Support for the ND2 format will be unaffected and continue via this newly renamed reader
- **Legacy QuickTime**
  - The LegacyQT Reader, Writer and Tools have been marked as deprecated in preparation for removal in the upcoming 7.0.0 major release of Bio-Formats. These classes depended on the external QuickTime for Java library which is long obsolete. From Bio-Formats 7.0.0 onwards the existing NativeQTReader will be renamed as QTReader and support for QuickTime will be limited to non legacy readers and writers.
- **LuraWave Codec**
  - The LuraWave Codec along with the LuraWaveService and associated components have been marked as deprecated in preparation for removal in the upcoming 7.0.0 major release of Bio-Formats. This codec depends on a third-party proprietary library with a license code. Opera Flex is the format that is most likely to be impacted by this change.
- **Woolz**
  - The Woolz Reader, Writer and Service have been marked as deprecated in preparation for removal in the upcoming 7.0.0 major release of Bio-Formats. The reader and writer are untested and rely on an underlying library which maintenance status is unclear

File format fixes and improvements:

- **CV7000/CV8000**
  - updated CV7000 isThisType test to improve performance by skipping extended type checking
  - reader now handles the use case where wells are recorded, but all files were removed
  - fixed channel indexing in scenarios where more channels are defined than acquired
- **ICS (Image Cytometry Standard)**
  - fixed an Illegal group reference exception when constructing slice label
- **KLB (Keller Lab Block)**
  - fixed a bug with the reading of image planes in Z stacks
- **MicroManager**
  - updated handling of truncated files to prevent possible infinite loop and throw an exception

Bio-Formats improvements:

- updated handling of exceptions in Bio-Formats plugins to ensure readers are closed
- updated list of external readers to include new external ZeissQuickStartCZIReader (thanks to Nicolas Chiaruttini)
- updated top-level Git mailmap to normalize commit author variants

- added a new bf-unconfigured tool to the Bio-Formats toolbox. The new tool lists classes which are not configured for testing as part of format reader tests
- replaced use tt tag in java docs with code tag

Documentation improvements:

- fixed a number of broken links
- migrated OME-Model documentation to Read the Docs
- added deprecation warnings to format pages of the affected components

Component updates:

- *ome-common* was upgraded to 6.0.17

5.16 6.13.0 (2023 April)

File format fixes and improvements:

- **Aperio SVS**
  - removed broken special case path when there is only one (pyramidal) image (thanks to Alexandr Virodov)
- **Bio-Rad Gel**
  - fixed pixel data offsets for cropped images
- **cellSens VSI**
  - improved tile decoding to fail faster on invalid tiles
- **CellWorX/MetaXpress**
  - fixed a bug when retrieving channel metadata when the files for the entire first field are missing
- **CV7000**
  - extra folders mixed in with the plate data will now be ignored
  - corrected channel indexing if a subset of channels was acquired
- **Lambert Instruments FLIM**
  - added support for Fli 2.0 (thanks to Shaquille Louisa - Lambert Instruments)
- **MetaMorph**
  - prevented an exception when attempting to read metadata keys that don't exist
- **MetaMorph TIFF**
  - fixed a NullPointerException when there are missing stage positions
- **MRC**
  - the IMOD flag is now used to determine int8 or uint8 pixel type
- **NIFTI**
  - improved the performance for gzip-compressed files by reducing the number of backwards seeks
- **Nikon ND2**
  - fixed an exception when reading blocks greater than 2GB
- **OME-TIFF**

  - file paths are now normalized rather than resolving symlinks

- **Perkin Elmer Operetta**
  - fixed series validation for single plane images (thanks to Alex Herbert)
- **Zeiss CZI**
  - improved the tile stitching when a pyramid does not exist

Bio-Formats improvements:

- added a new writer option *preserve_creator* for OME-TIFF and OME-XML writers. When set to true the original OME/Creator attribute will be preserved by the writer. If set to false, the OME Creator attribute will be updated to include the current Bio-Formats version. Attribute is set to false by default.
- enabled the configuration of the fill value for undefined pixel values through new API additions to IFormatReader. Two new methods (along with default implementations) have been added for *setFillColor(Byte color)* and *Byte getFillColor()*, the default value if undefined is 0.
- added a new *-fill* option to the command line tools, allowing a byte value to be configured for use with undefined pixels (0-255)
- enabled the configuration of JPEG quality in the JPEGCodec using the existing quality setting on *CodecOptions* (thanks to Peter Haub)
- improved JPEG performance by using in-memory cache in JPEGCodec (thanks to Pete Bankhead)
- made minor API updates to *Memoizer* to allow a null reader to be specified in the constructors, in which case a new *ImageReader* will be wrapped
- prevented file leaks in FileStitcher by closing readers when setting the class list
- fixed exceptions in FileStitcher when a no-op file pattern is encountered
- updated GitHub Actions release process

Documentation improvements:

- updated the format page for MRC to clarify list of developers and add links to latest specifications
- added new developer documentation pages for Bioconductor RBioFormats
- corrected one of the examples in the OME-TIFF specification page (thanks to Tom Boissonnet)
- updated the list of writer options to add documentation for new *preserve_creator* options
- updated the list of command line tools options to add documentation for new *fill* option
- version history and bug reporting are now linked from the main documentation page
- fixed a number of broken links
- updated the Li-Flim format page to add supported versions

Component updates:

- *ome-common* was upgraded to 6.0.16
- *ome-model* was upgraded to 6.3.3
- *ome-codecs* was upgraded to 0.4.5
- *imagej1* was upgraded to 1.54c
- *json* was upgraded to 20230227
- *kryo* was upgraded to 5.4.0

- *snakeyaml* was upgraded to 2.0
- *joda-time* was upgraded to 2.12.2
- *guava* was upgraded to 31.1-jre
- *jackson-annotations* was upgraded to 12.14.2
- *jackson-databind* was upgraded to 12.14.2

5.17 6.12.0 (2023 February)

File format fixes and improvements:

- **Lambert Instruments FLIM**
  - added support for MSB 12 to 16 bit conversion (thanks to Shaquille Louisa - Lambert Instruments)
- **Olympus cellSens VSI**
  - corrected tile handling to respect tile origin metadata
  - pixel data will now only be read from ets files with the prefix *frame_* (funded by a partnership between Glencoe Software and Evident Scientific)
- **Perkin Elmer Operetta/Harmony**
  - plates containing 32-bit TIFF files will now return blank planes
- **Zeiss CZI**
  - fixed pyramid dimension calculation

Bio-Formats improvements:

- implemented full Java 17 support for Bio-Formats
- **logback dependencies are now optional as per the SLF4J best practices (thanks to Curtis Rueden)**
  - downstream consumers must now declare the specific SLF4J binding they wish to use
- CI build matrix updated to use Java versions 8, 11 and 17
- ant tasks switched to use Maven Artifact Resolver
- fixed a bug when decompressing using the Base64Codec
- added a new ImageJ macro example to Bio-Formats examples
- FakeReader now ensures correct dimension order for RGB image
- added new unit tests for FakeReader
- fixed min/max calculation for RGB data
- artifacts from the GitHub Actions build are now uploaded and archived for 30 days
- release artifacts are now generated via GitHub Actions
- fixed a number of small issues in OME-Model (thanks to Roger Leigh)

Documentation improvements:

- migrated Bio-Formats documentation to Read the Docs
- fixed and updated a number of links in documentation
- updated OME-Model links to comply with Sphinx 5 deprecation warnings

- updated GitHub Actions set-output to comply with deprecation warnings
- updated OME-TIFF documentation to add links to new sample dataset BBBC017

Component updates:

- *ome-metakit* was upgraded to 5.3.5
- *ome-common* was upgraded to 6.0.14
- *ome-codecs* was upgraded to 0.4.4
- *ome-model* was upgraded to 6.3.2
- *jackson-annotations* was upgraded to 12.14.1
- *jackson-databind* was upgraded to 12.14.1
- *aircompressor* was upgraded to 0.21

5.18 6.11.1 (2022 December)

File format fixes and improvements:

- **3i SlideBook 7**
  - refactored code to remove a false positive threat report in Fortinet
- **DICOM**
  - fixed handling of some datasets with multiple optical paths
- **Leica SCN**
  - fixed colour correction for datasets from SCN 400 models
- **Olympus cellSens VSI**
  - corrected the handling of exposure times
- **TIFF**
  - handle REFERENCE_BLACK_WHITE tag as an array of floats or ints
- **Vectra QPTIFF**
  - biomarker data in channel name will now be preserved
- **Zeiss CZI**
  - added support for plates with multiple fields

Bio-Formats improvements:

- updated the output command for GitHub actions

Documentation improvements:

- fixed a number of broken links
- added a link to public sample files for Olympus-FluoView

Component updates:

- *snakeyaml* was upgraded to 1.3.2

5.19 6.11.0 (2022 October)

File format fixes and improvements:

- **Amira Mesh**
  - fixed handling of files with empty key values
- **InCell 1000/2000**
  - improved calculation of plane count
- **Nikon NIS-Elements ND2**
  - fixed handling of chunk map skipping logic
- **Olympus FluoView FV1000**
  - fixed an exception when parsing double values in the metadata
- **OME-TIFF**
  - improved initialization performance for files with a large number of planes
  - enabled individual file reading for datasets with one file per series
- **SimplePCI & HCImage**
  - fixed physical size calculation to make use of magnification value
- **TIFF**
  - implemented Zstandard decompression for Tiff formats (thanks to Willem Pomp)

Bio-Formats improvements:

- updated pattern file handling to support a single file wrapped in a pattern file
- Bio-Formats plugin now closes underlying readers after an exception
- contrast will be retained when using VirtualImagePlus in the Bio-Formats plugin (thanks to Tomas Farago)
- fixed overwrite checking in bfconvert tool when output path is a pattern

Documentation improvements:

- fixed a number of broken links
- updated test image documentation to add note on on setting fake file channel colors
- added new public sample files for Olympus-FluoView
- updated OME-TIFF sample files for BBBC017

Component updates:

- *ome-codecs* was upgraded to 0.4.1
- *snakeyaml* was upgraded to 1.3.1

5.20 6.10.1 (2022 August)

File format fixes and improvements:

- **Hamamatsu NDPI**
  - fixed handling of non number tags for files greater than 4GB in size
- **OME-TIFF**

  - improved performance of initializing a single file OME-TIFF datasets with a larger number of series

- **TIFF (Tagged Image File Format)**
  - fixed a ClassCastException when the SubfileType tag has the wrong type
- **Zeiss CZI**
  - scene names will now be taken into account when generating image names

Documentation improvements:

- fixed a number of broken links

Component updates:

- *ome-common* was upgraded to 6.0.13
- *ome-poi* was upgraded to 5.3.7
- *slf4j* was upgraded to 1.7.30
- *metadata-extractor* was upgraded to 2.18.0

5.21 6.10.0 (2022 May)

New file formats:

- **Slidebook 7**
  - added support for reading SlideBook Format 7 from SlideBook 2021 This functionality was implemented and contributed by Intelligent Imaging Innovations

File format fixes and improvements:

- **CellWorX / MetaXpress**
  - channel metadata will now be preserved for all channels by reading from each file
- **Cellomics**
  - marked the private Pattern field as transient to fix memo file generation with JDK17
- **DeltaVision**
  - fixed detection of incorrect XY tile counts
- **Hitachi S-4800**
  - enhanced format recognition and identification of datasets
  - corrected the units for physical sizes to nanometers instead of micrometers
- **Nikon NIS-Elements ND2**
  - improved metadata parsing, especially for objective data, timestamps, and exposure times
- **PerkinElmer Operetta**
  - improved checks for invalid TIFFs and supplemental metadata files
  - updated image names to make well names more readable
- **TIFF (Tagged Image File Format)**
  - fixed incorrect SampleValue metadata values
- **Vectra QPTIFF**

  - reader has been updated to support schema version 4
  - funded by a partnership between Glencoe Software and Akoya Biosciences.

Bio-Formats improvements:

- updated automated testing to ensure files are initialized before all tests

Documentation improvements:

- fixed broken link in MIPAV documentation
- updated logback component version in developer docs
- updated link to View5D software
- added a new format page for Slidebook 7

Component updates:

- *ome-metakit* was upgraded to 5.3.4
- *ome-common* was upgraded to 6.0.9
- *ome-model* was upgraded to 6.3.1
- *ome-poi* was upgraded to 5.3.6
- *ome-codecs* was upgraded to 0.3.2
- *logback-core* was upgraded to 1.2.9
- *logback-classic* was upgraded to 1.2.9
- *xercesImpl* was upgraded to 2.12.2
- *xml-apis* was upgraded to 1.4.01
- *snakeyaml* 1.29 was added as a dependency

5.22 6.9.1 (2022 April)

File format fixes and improvements:

- **DeltaVision**
  - allowed partial planes to be read from truncated files
- **MetaMorph**
  - fixed a NumberFormatException when parsing double values in metdata
- **OME-TIFF**
  - performance improvements to reduce the number of open file handles during initialization
- **PerkinElmer Operetta**
  - performance improvements to speed up the reading of datasets (thanks to Nicolas Chiaruttini)
- **Zeiss CZI**
  - switched to a white background for brightfield data to better match Zeiss Zen software
- **Zeiss LSM**
  - fixed a bug that resulted in an incorrect pixel type for some floating point data

Bio-Formats improvements:

- removed the maven deploy step from workflows in forked repositories
- upgraded cdm dependency from 4.6.13 to cdm-core 5.3.3
- configurable sleep time in FakeReader moved to the end of initialization

Documentation improvements:

- DICOM format page updated with improved links for sample datasets, software, and specification
- updated guidelines for submitting sample datasets

5.23 6.9.0 (2022 February)

New file formats:

- **Leica LOF**
  - added support for reading Leica LOF files. This functionality was implemented and contributed by Leica Microsystems
- **Leica XLEF**
  - added support for reading Leica XLEF files. This functionality was implemented and contributed by Leica Microsystems

File format fixes and improvements:

- **FEI TIFF**
  - fixed parsing of physical pixel sizes for Phenom data
- **Imspector OBF**
  - improved handling of deflate errors when opening older OBF files (thanks to Nils Gladitz)
- **JPEG**
  - performance improvements to reduce memory required to read tiles from large JPEGs
- **Leica LIF**
  - improved parsing of channel metadata (thanks to Zach Marin)
- **Nikon NIS-Elements ND2**
  -improved parsing of metadata tables with invalid characters
- **OME-TIFF**
  - performance improvements of tile read speeds for some pyramid OME-TIFFs
- **PerkinElmer Operetta**
  - enabled support for handling sparse planes

Bio-Formats improvements:

- added new API methods to FormatTools for creating well names
- added a swap option to bfconvert to override input dimension order (thanks to Roberto Calabrese)

Documentation improvements:

- new public sample files for Leica XLEF (thanks to Leica Microsystems)
- added documentation for using the swap option with the command line tools
- updated the process for contributing sample files via Zenodo

- updated the link to NDP.view2 software on the Hamamatsu ndpi format page

5.24 6.8.1 (2022 January)

File format fixes and improvements:

- **Aperio SVS / Aperio AFI**
  - improved handling of macro and label images when no image description is present
- **cellSens VSI**
  - added support for physicalSizeZ metadata
- **Gatan Digital Micrograph DM4**
  - fixed a FormatException when encountering null values in the metadata

Bio-Formats improvements:

- fixed a bug in TiffSaver which could result in an invalid TIFF when saving a file with tiling and compression (thanks to Pete Bankhead)
- made performance improvements to TIFF reading and writing (thanks to Pete Bankhead)

Security improvements:

- Updated use of the git:// protocol in POM as it has now been deprecated. See the GitHub blog post for further details
- Removed the loci_tools from Bio-Formats builds due to log4j vulnerability, users should instead use bioformats_package. See the OME security advisory for further details

5.25 6.8.0 (2021 December)

New file formats:

- **DICOM**
  - added support for reading and writing DICOM whole slide images (DICOM WSI format). This functionality was implemented through collaboration with NCI Imaging Data Commons, and has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, under Task Order No. HHSN26110071 under Contract No. HHSN2612015000031
- **Olympus omp2info**
  - Added a new reader for the Olympus tile format provided through a partnership with Glencoe Software and OLYMPUS EUROPA SE & Co. KG

File format fixes and improvements:

- **Amira Mesh**
  - added support for Amira 3.0 keys
- **Aperio SVS / Aperio AFI**
  - improved handling of macro and label images
- **AVI (Audio Video Interleave)**
  - set fps from Pixels Time Increment when writing AVI files
- **Nikon NIS-Elements ND2**

  - updated parsing of newer ND2 files to resolve issues with incorrect dimensions(thanks to Ilya Parmon)

- **Olympus OIR**
  - fixed a Null Pointer Exception for Laser Data ID
- **OME-TIFF**
  - fixed handling of partial datasets
  - added a new reader option ometiff.fail_on_missing_tiff to configure behaviour of partial dataset
- **PerkinElmer Operetta**
  - added support for Phenix v6 data
- **TIFF**
  - fixed non-sequential offset correction for TIFF files between 2 and 4 GB
- **Zeiss CZI**
  - added support for zstd compression. This feature will require FIJI users to additionally download the dependency io.airlift.aircompressor 0.18. (funded by a partnership between Glencoe Software and ZEISS)

Bio-Formats improvements:

- updated the ordering of reader.txt
- *bfconvert* tool updated to use multiple of tile size granularity when writing (thanks to Jeremy Muhlich)
- added a new no-sequential option to enable writing in non sequential order

Component updates:

- *jhdf5* was upgraded to 19.04.0
- *commons-lang* 2.6 was added as a dependency
- *aircompressor* 0.18 was added as a dependency

Documentation improvements:

- updated page for using Bio-Formats in Python to add links for AICSImageIO and PyImageJ (thanks to Curtis Rueden)

5.26 6.7.0 (2021 August)

File format fixes and improvements:

- **cellSens VSI**
  - corrected offset checking to prevent seeking beyond EOF
- **Deltavision**
  - fixed a bug which resulted in an IndexOutOfBoundsException
- **Hamamatsu ndpis**
  - added support for channel names from NDP Shading Data
- **Nikon NIS-Elements ND2**
  - improved parsing of timestamp values (thanks to Ilya Parmon)
  - improved parsing of channel names and colors

- **Olympus FluoView FV1000**
  - fixed a bug with the handling of filter indexes
- **OME-TIFF**
  - resolution annotations now removed when resolutions are flattened
  - fixed handling of filesets with BinaryOnly across multiple folders
- **PerkinElmer Operetta**
  - channel colors now populated using emission wavelength
- **TIFF**
  - added support for Deflate compressed tiles/strips with lsb2msb order
  - prioritised units from TIFF tag over those from ImageJ comment
- **Zeiss CZI**
  - plates will now be correctly detected
  - added a new reader option `zeissczi.relative_positions` which when set to true will change the PositionX and PositionY values stored in OME-XML to the pixel position instead of the absolute physical stage position
  - added a new reader option `zeissczi.trim_dimensions` which when set to true will use the pixel block metadata to trim XY dimensions to match those reported in ZEN

Bio-Formats improvements:

- the pattern reader now supports populating channel names from pattern tokens
- the channel filler will correctly reset bits per pixel if a lookup table is applied
- existing reader options are now registered in getAvailableOptions command
- improved performance of OME-XML validation (thanks to Nils Gladitz)
- added unit tests for the upgrade of custom attributes in OME-XML transforms

Documentation improvements:

- updated link to a 2005 publication about the OME data model
- bumped low level components *logback-core* and *logback-classic* to 1.2.0
- updated the licensing for BDV, KLB and CellH5 readers to BSD
- added documentation for additional reader options
- fixed a number of broken links

The below have been relicensed under the more permissive BSD-2 clause:

- Big Data Viewer reader
- Keller Lab Block reader
- CellH5 reader
- JHDF service
- Bio-Formats GNU Octave package
- Bio-Formats MATLAB functions

Component updates:

- *ome-model* was upgraded to 6.2.3
- *ome-common* was upgraded to 6.0.7
- *ome-poi* was upgraded to 5.3.4
- *ome-codecs* was upgraded to 0.3.1
- *ome-metakit* was upgraded to 5.3.3
- *logback* was upgraded to 1.2.0

5.27 6.6.1 (2021 March)

File format fixes and improvements:

- **CV7000**
  - fixed the channel-wise ordering of planes. The implementation changes to channel mapping will require existing memo files to be regenerated for CV7000 datasets
- **Hamamatsu NDPIS**
  - transmittance values will now be used to pick valid channels if no wavelength present
- **Leica SCN**
  - updated position units from reference frame to nm
  - original metadata now populated correctly for all series
- **MetaMorph Stack**
  - support added for parsing *NDInfoFile Version* to determined correct file suffix
  - improved detection of companion binary files
- **MetaXpress**
  - improved detection of thumbnail paths
- **Nikon NIS-Elements ND2**
  - made performance improvements to reduce memory usage when reading large datasets
- **Olympus OIR**
  - fixed a bug which resulted in blank planes when XML blocks end with CRLF

Documentation improvements:

- updated *bfconvert* documentation for using pattern string on Windows (thanks to Nathanael Reveal)
- added a Fiji usage note to the Tecan Spark Cyto Workspace format page
- added references to the public Imaris IMS format specification page

Bio-Formats improvements:

- added support for *ChannelName* in Fake series tables

5.28 6.6.0 (2020 December)

New file formats:

- **Tecan Spark Cyto Workspace**

  - Added a new reader for Tecan Spark Cyto workspace files, provided through a collaboration between Tecan Trading and Glencoe Software

File format fixes and improvements:

- **Applied Precision CellWorX / MetaXpress**
  - support for CellWorX and MetaXpress has now been split into separate readers
- **BD Pathway**
  - plate row and column dimensions now being populated
- **Bitmap**
  - fixed offset calculation for files larger than 2 GB
- **Cellomics**
  - channel data now being parsed from companion .mdb file
  - corrected Plate/Well/Image mappings for sparse plates
  - plate size now calculated using the maximum row/column index
  - plate row and column dimensions now being populated
- **Gatan Digital Micrograph DM4**
  - fixed support for montages with single Z dimensions
- **MetaMorph Stack**
  - corrected laser indexes for multi series datasets
- **MIAS (Maia Scientific)**
  - plate row and column dimensions now being populated
- **Nikon NIS-Elements ND2**
  - fixed a null pointer exception when parsing metadata key value pairs
- **Olympus ScanR**
  - added functionality to handle missing wells through a new option `scanr.skip_missing_wells`. By default the option is set to true and missing wells are skipped
- **Olympus SIS TIFF**
  - corrected parsing of pixel size values (thanks to Stephan Wagner-Conrad)
  - removed trailing null byte from imageName and channelName
  - improved formatting of the image reader
- **PerkinElmer Opera Flex**
  - plate row and column dimensions now being populated
- **PerkinElmer Operetta**
  - updated metadata files logic to skip plate folders
- **PNG (Portable Network Graphics)**
  - fixed an issue which resulted in a hanging call to *openBytes*
- **Zeiss CZI**
  - scene number is now correctly padded and indexes begin at 1

- **Zeiss LSM**
  - improved handling of cached plane variables

Documentation improvements:

- added new OME-TIFF plate companion sample files
- references to image index in the API documentation have been updated to plane index
- updated various links to follow the LOCI site migration
- fixed broken Javadoc links
- removed outdated references to mailing lists

Component updates:

- *ome-model* was upgraded to 6.2.2
- *ome-common* was upgraded to 6.0.6
- Memoizer version has been incremented meaning previous memo files are invalidated and will be regenerated
- *kryo* dependency updated to 4.0.2
- Added new *sqlite-jdbc* version 3.28.0 dependency for the Tecan Spark Cyto Workspace format

Bio-Formats improvements:

- removed automatic file stitching from format reader tests
- improved stringency of ChannelName and ImageName testing
- improved reader detection for image conversion testing
- reviewed all instances of whitelist/blacklist
- introduced support for GitHub Actions
- improved handling of DynamicMetadataOptions on Windows
- introduced the ability to set metadata options using a *.bfoptions* file
- test-suite updated to handle new *.bfoptions* file

5.29 6.5.1 (2020 July)

File format fixes and improvements:

- **Aperio SVS / Aperio AFI**
  - fixed a Null Pointer Exception when exposure time is not defined
- **Big Data Viewer**
  - corrected series indexes for non flattened multi resolution images
- **Cellomics**
  - physical sizes are now set for all series rather than just the first
- **Imspector OBF**
  - file format version and stack version are now recorded as part of global metadata
- **MetaMorph**
  - improved wavelength parsing using Metamorph XML or original metadata

- **Mikroscan TIFF**
  - stricter format recognition now used to prevent erroneous use of the reader
- **Ventana BIF**
  - added support for LEFT overlap direction (thanks to Joan Gibert)
- **Zeiss CZI**
  - fixed a bug to ensure Channel Illumination Type is not overridden by display settings

Documentation improvements:

- added link from OME-TIFF page to commercial partners page
- updated links for Biplane to now use Oxford Instruments
- fixed a number of broken hyperlinks in documentation

Component updates:

- *jxrlib* was upgraded to 0.2.4

Bio-Formats improvements:

- fixed a bug in *bfconvert* for multi-series files with varying image sizes
- removed the logging OMERO IDs passed to FormatReader and ImageReader

5.30 6.5.0 (2020 April)

File format fixes and improvements:

- **Big Data Viewer**
  - improved performance of tiled reading
- **DeltaVision**
  - implemented additional sanity check to header to ensure correct panel count
- **DICOM**
  - improved performance of initialization of multi-file datasets. Files spread across multiple directories are now handled by a DICOMDIR file that groups the dataset
- **Imspector OBF**
  - added support for OBF Version 6 stacks (thanks to Nils Gladitz)
  - added support for OBF Version 4 stack flush points (thanks to Nils Gladitz)
- **MetaMorph**
  - ensured dimension metadata read from tags and/or .nd file rather than allowing the underlying TIFF reader to treat each IFD as a series
- **MetaXpress**
  - added support for single site HCS variant
- **Zeiss CZI**
  - fixed the position count when only one position is present and the starting index is greater than 0
- **Various Readers**
  - reviewed and updated readers to prevent potential cases of integer overflow

OME-Model updates:

- version of `OME-Model` has been updated to 6.1.0
- added support for Python 3.8 and make code-generation Python 3 only (thanks to Roger Leigh)
- added getters and setters for OME@Creator attribute for ome.xml metadata interfaces and implementations (thanks to Nils Gladitz)
- removed unmaintained C++ OME-XML implementation (thanks to Roger Leigh)
- updated code-generation for building on Python 3.6, 3.7 (thanks to Roger Leigh)
- uncapped the Sphinx version for OME-Model documentation

Documentation improvements:

- fixed a number of broken links within the documentation
- added a new format page for MetaXpress
- added a reference to public OBF sample images in the format page

5.31 6.4.0 (2020 March)

File format fixes and improvements:

- **Applied Precision CellWorX**
  - added support for multiple Z sections
- **DeltaVision**
  - added and updated objective metadata based on values from softWoRx 7.2.0 (thanks to David Pinto)
- **Hamamatsu NDPI**
  - added support for JPEG-XR compression
  - added full support for files larger than 4 GB
  - improved support for a number of additional metadata tags
  - the Hamamatsu NDPI reader improvements are provided via work from Glencoe Software Inc.
- **InCell**
  - inverted Y coordinate in plane/field positions to correct stitching of tiles
- **PerkinElmer Vectra QPTIFF**
  - plane position values will now be populated on OME-XML
- **TIFF**
  - values for XPosition and YPosition in original metadata will now be more accurately stored as doubles
  - implemented a fix to prevent integer overflow when reading from a large tile greater than 2 GB
- **Ventana BIF**
  - improved handling of physical sizes for pre-stitched TIFFs
- **Zeiss CZI**
  - added a fix for uncompressed pixels incorrectly flagged as JPEG-XR
  - fixed a bug so that line-scans are now read correctly (thanks to Stephan Wagner-Conrad)

  - improved parsing of detector metadata

Bio-Formats tools improvements:

- added a new `nobigtiff` option to `bfconvert` to disable automatic switching to BigTiff based upon the number of pixel bytes (TIFF files larger than 4GB). This may be useful when converting using a compression codec so that the output file size is less than 4GB
- fixed a bug in `xmlvalid` tool to properly handle lowercasing of file names
- added new bfGetPlaneAtZCT function to MATLAB toolbox to retrieve a particular plane at a ZCT coordinate (thanks to Mark Kittisopikul)
- added a new bfTestInRange helper function to MATLAB toolbox with improved performance and error handling (thanks to Mark Kittisopikul)
- fixed a bug when using `bfconvert` on multi-series files with only a single timepoint, channel or Z slice selected

Bio-Formats API updates:

- version of `jxrlib` has been updated to 0.2.2
- version of `ome-codecs` has been updated to 0.3.0 which includes performance upgrades for LZW compression (thanks to Alexander Popiel)
- moved JPEG-XR codec and service from `formats-gpl` to `formats-bsd` component
- TiffParser and TiffSaver have now been updated to implement Closeable
- added a documentation note to use one IFD instance per plane with `saveBytes` in `TiffWriter`
- FormatWriter will now create output file's parent directory if needed
- FakeReader now allows for DeltaT to be set in INI file
- FakeReader now handles INI files in plates created by `mkfake`
- fixed a number of deprecation warnings in various readers

5.32 6.3.1 (2019 December)

File format fixes and improvements:

- **ICS (Image Cytometry Standard)**
  - prevented a potential error when writing ICS files with physical units which could not be converted
- **Imspector OBF**
  - fixed a bug with incorrect dimensions being parsed for some Imspector OBF files
- **Leica LAS AF LIF (Leica Image File Format)**
  - fixed a NullPointerException in some variants of the LIF file format
- **TIFF**
  - improved the performance of tiled writing
- **Zeiss CZI**
  - fixed issues with tile stitching and position size

Bio-Formats tools improvements:

- fixed a potential NullPointerException in SpringUtilities for Bio-Formats plugins (thanks to July Chen)
- updated URL for fetching ImageJ upgrades in `ijview`

- fixed the XY coordinates for cropped images in `bfconvert`
- fixed a bug when using a cropped multi-series file in `bfconvert` (thanks to Matthieu Moisse)
- fixed issues in `bfconvert` when writing separate tiles with additional options
- added documentation of tile output patterns to utility help in `bfconvert`

5.33 6.3.0 (2019 October)

File format fixes and improvements:

- **Big Data Viewer**
  - added support for parsing of physical sizes
- **DeltaVision**
  - added a new RCPNL reader which is a variant and split out of the DeltaVision format
- **Hamamatsu NDPI**
  - fixed population of the nominal magnification from the SourceLens TIFF tag
- **ICS (Image Cytometry Standard)**
  - fixed a bug when using tiles to read files from SVI-Huygens
- **Imspector OBF**
  - fixed an *IndexOutOfBoundsException* exception when using *DummyMetadata* (thanks to Nils Gladitz)
- **JPEG 2000**
  - added support for sub-resolutions
- **Leica LIF**
  - updated to parse attachments to determine if XY positions should be flipped or swapped
- **MetaMorph**
  - improved file name construction and plate detection logic
- **Nikon ND2**
  - updated to use floating point for 32 bit values
- **OME-TIFF**
  - reduced memory usage when reading files and memo file size for cached files
- **PerkinElmer Operetta**
  - improved handling of empty fields to prevent series from having X or Y set to 0

API updates:

- added *overwriteIFDValue* signature that takes an IFD offset to *TiffSaver*
- added a new *getRequiredDirectories* method to *FormatTools*
- new *FakeReader* keys added for *sleepOpenBytes* and *sleepInitFile*

Build updates:

- updated deployment mechanism for SNAPSHOT and Release to use Travis CI
- increased the strictness of AcquisitionDate checks in *FormatReaderTest*

- improved test coverage of companion file datasets

Component updates:

- *ome-common* was upgraded to 6.0.4
- updated *DateTools* to attempt to parse invalid dates with *Locale.US*
- *DateTools* documentation updated to clarify expected units for timestamp passed to convertDate

Documentation improvements:

- added documentation for sleep options when generating test images
- fixed broken external links in documentation
- corrected suffixes used for JPEG 2000

5.34 6.2.1 (2019 August)

File format fixes and improvements:

- **Applied Precision CellWorX**
  - corrected plane positions for series index > 0 rather than reusing positions from the first series
- **DeltaVision**
  - added objective info for new Applied Precision 100X/1.4 lensID
  - updated so that date from dv file will override log file date to avoid locale-dependent dates
- **Leica LAS AF LIF (Leica Image File Format)**
  - fixed units and indexing for tile-based plane positions
- **TIFF**
  - fixed a potential exception in MinimalTiffReader when the TIFF is stored using very large tile/strip dimensions
- **Zeiss CZI**
  - fixed a potential index out of bounds exception when populating positions

Bug fixes and improvements:

- *bfconvert* has been updated so that when the dimensions of a sub-resolution are smaller than the requested tile size then they default to the size of the sub-resolution
- fixed a bug in the execution of *bfsave* in the GNU Octave environment

Codec updates:

- *ome-codecs* was upgraded to 0.2.5
- JPEG codec updated to reduce decompression time for 8-bit RGB images
- Huffman codec updated to allow the decoding tree to go all the way down to the 16-bit depth required by the standard (thanks to Aaron Avery)
- Lossless JPEG codec updated to provide better compliance with the LJPEG standard (thanks to Aaron Avery)

Documentation improvements:

- added instructions for building Bio-Formats with IntelliJ IDEA
- corrected command-line tools documentation for the novalid and noncore options

- updated broken links to Barre's Medical Imaging Samples
- updated the imagej.net link for Zeiss LSM toolbox plugin
- **added links to public sample files for the following formats:**
  - Big Data Viewer
  - CellWorX
  - CellH5
  - PerkinElmer Opera Flex
  - Gatan DM3
  - Image Cytometry Standard
  - Keller Lab Block
  - PerkinElmer Columbus
  - Ventana BIF
  - Zeiss-CZI

5.35 6.2.0 (2019 July)

New file formats:

- **Mikroscan TIFF**
  - a new reader for Mikroscan TIFF files has been contributed with thanks to Jim Crowe, Mikroscan Technologies, Inc.
- **Ventana BIF**
  - added a new reader for Ventana BIF files which has been commissioned via Glencoe Software

File format fixes and improvements:

- **Cellomics**
  - fixed indexing for plates with a single well or missing fields
- **DeltaVision**
  - added support for the reading of the new panel count field (provided through a collaboration between GE Healthcare and Glencoe Software Inc.)
- **PerkinElmer Operetta**
  - images with smaller XY dimensions than all other TIFF files in dataset will now be padded
- **TIFF**
  - updated functionality for overwriting IFD values to ensure that previous value is completely overwritten and no orphaned tags are left
- **Zeiss CZI**
  - expanded support for auto-stitching of tiles

Bug fixes and improvements:

- added `-cache`, `-cache-dir` and `-no-sas` options to `bfconvert` tool
- deprecated broken TRUNK and DAILY builds from upgrade checker

- disabled Oracle JDK from Travis CI checks

Documentation improvements:

- fixed broken link for discontinued Dcraw software
- updated links for Zeiss formats

5.36 6.1.1 (2019 June)

File format fixes and improvements:

- **DeltaVision**
  - added new lens definitions associated with *rcpnl* files
- **Gatan Digital Micrograph (DM3/DM4)**
  - now parsing the *Montage* tag to determine if tiles are present
- **Leica LAS AF LIF (Leica Image File Format)**
  - added fix to correctly read scale from polygon regions of interest (thanks to Sean Warren)
- **PerkinElmer Columbus**
  - improved handling of truncated TIFF files to return blank planes
- **PerkinElmer Opera Flex**
  - plate barcodes are now used to improve grouping and handling of truncated files
- **TIFF (Tagged Image File Format)**
  - improved parsing times for images stored as uncompressed contiguous strips
- **Zeiss CZI**
  - improved plane position metadata for many CZI datasets

Automated test changes:

- format reader tests have been updated to handle PerkinElmer Columbus datasets with flex files

Documentation improvements:

- added help for missing options in `bfconvert` command line tool

5.37 6.1.0 (2019 May)

New file formats:

- **BDV**
  - added a new reader for Big Data Viewer files

File format fixes and improvements:

- **Applied Precision CellWorX**
  - improved handling of thumbnail files
- **DeltaVision**
  - updated handling of *rcpnl* files to treat each file as a single timepoint
- **FakeReader**

  - removed *header* key from original metadata
- **Hamamatsu VMS**
  - removed *header* key from original metadata
- **Hitachi S-4800**
  - removed *header* key from original metadata
- **ICS (Image Cytometry Standard)**
  - fixed an issue reading .ics/.ids files written by SVI Huygens (thanks to Jan Eglinger)
- **Imaris IMS**
  - fixed issues with newer files which had been failing due to older *netcdf* version
- **JPEG**
  - improved the reading of EXIF data
- **Lambert Instruments FLIM**
  - added support for packed UINT12 datatype (thanks to Johan Herz)
- **LEO**
  - fixed a bug with the parsing of physical sizes
  - improved support for additional global metadata fields
- **Olympus OIR**
  - fixed a bug which would show empty pixels when more than 1000 timepoints

Automated test changes:

- added additional tests for HCS/SPW datasets to ensure Plate, PlateAcquisition, Well, WellSample, and WellSample position values are configured where present
- added a new *file-leak-detector* test to flag potential memory leaks

Bio-Formats API changes:

- `ImageConverter` as used in `bfconvert` command line tool is now public
- made `ImageReader` more defensive against exceptions thrown when determining reader type
- fixed an issue when performing a non-sequential write for multi-resolution TIFF files

Component changes:

- *ome-common* was upgraded to 6.0.3
- *perf4j* was upgraded to 0.9.16
- removed *Guava* dependency which will be pulled transitively from the upstream *ome-common* dependency
- *jhdf5* was upgraded to 14.12.6
- *metadata-extractor* was upgraded to 2.11.0
- *xercesImpl* version 2.8.1 was added as it is no longer a dependency of *metadata-extractor*
- *netcdf* was upgraded to 4.6.13

5.38 6.0.1 (2019 March)

File format fixes and improvements:

- **cellSens VSI**
  - improved tag parsing resulting in fixes for missing or incorrect metadata
- **Hamamatsu ndpi**
  - improved handling of variants where a constituent NDPI has no wavelength
- **LaVision Imspector**
  - fixed a potential NullPointerException when 'xyz-Table Z Resolution' is false
- **NRRD (Nearly Raw Raster Data)**
  - added support for raw GZIP-compressed data files
- **Olympus OIR**
  - fix to ensure file path is normalized which fixes detection on Windows
- **TIFF**
  - improved handling of direct tile copying to prevent invalid images
  - improved handling of tiles in scenarios of an invalid offset or byte count of 0

Documentation improvements:

- added documentation for -noflat option to the showinf and bfconvert users pages
- updated recommended minimal MATLAB version to R2017b
- documented support for MATLAB versions prior to R2017b
- links to MicroCT public datasets now point to the public archive rather than directly to the zip file

5.39 6.0.0 (2019 February)

Bio-Formats API changes:

- Java 8 is now the minimum supported version
- Sub-resolution reading:
  - added `MetadataList` and `CoreMetadataList` classes
  - added a new `SubResolutionFormatReader` abstract class for handling pyramidal format readers
  - updated all pyramid format readers to use `SubResolutionFormatReader`
  - deprecated `getCoreMetadataList`, `seriesToCoreIndex`, `coreIndexToSeries`, `getCoreIndex` and `setCoreIndex` in `IFormatWriter`
- Added a new `IPyramidHandler` interface with the resolution getter methods
- Sub-resolution writing changes:
  - `IFormatWriter` now extends `IPyramidHandler` (breaking)
  - added `setResolutions` and `getResolutions` methods to `IFormatWriter` (breaking)
  - added examples of using the sub-resolution writing API
- Tiled writing API changes:

  - updated `IFormatWriter` to use `setTileSizeX(0)` and `setTileSizeY(0)` as a way to disable tiling (breaking)
  - updated `FormatWriter` set 0 as the default values of `getTileSizeX()` and `getTileSizeY` (breaking)
- `IFormatWriter.getCompressionTypes` now returns the types for the selected writer only
- Metadata handling:
  - added getter methods to `MetadataTools` for retrieving OME enumerations by value
  - deprecated OME enumeration getter methods in `FormatReader`
- Refactor `FilePatternReader` logic in a new `WrappedReader` abstract class

New file formats:

- KLB
  - added a new reader for Keller Lab Block (KLB) files
- CV7000
  - added a new reader for Yokogawa CV7000 datasets
- GE MicroCT
  - added a new reader for GE MicroCT datasets

File format fixes and improvements:

- Aperio SVS/AFI
  - removed pyramidal resolutions of mismatching pixel types
  - fixed exposure times, improved image naming of AFI datasets
  - displayed original metadata keys for each channel of AFI datasets
  - added support for multiple Z sections
- DICOM
  - improved file grouping and file-to-series mapping for multi-file datasets
- Fake
  - added support for multi-resolution test images
  - now populating WellSample positions when present using Plane data
- Gatan Digital Micrograph
  - adjusted endianness and record byte count for long values
  - allowed ROIs to be stored in DocumentObjectList groups
  - no longer creating an empty ROI when an unsupported shape type is encountered
- Image Pro
  - added support for Image Pro Plus .ips set
- GE InCell
  - added support for parsing minimum and maximum pixel values
- Lambert Instruments FLIM
  - fixed an integer overflow error with large files (thanks to Rolf Harkes)

- Leica LIF
  - unified metadata parsing to use `DataTools.parseDouble`
- Leica SCN
  - improved support for Versa datasets
- Micro-Manager
  - improved handling of very large `*_metadata.txt` files
  - prevented `NumberFormatException` for invalid double values
  - add support for parsing *ChannelColor* from `*_metadata.txt` files
- Metamorph
  - added support for multi-dimensional .scan dataset created from Scan Slide (thanks to Jeremy Muhlich)
- MRC (Medical Research Council)
  - fixed endian detection for old-style headers
- Nikon ND2
  - prevented integer overflow when reading chunkmaps from files larger than 2GB
  - fixed handling of duplicate and incomplete exposure time lists
  - fixed chunk map handling when CustomData blocks are between ImageDataSeqs
- OME-TIFF
  - added support for reading OME-TIFF with pyramidal resolutions stored as SubIFDs
  - added support for writing OME-TIFF with pyramidal resolutions
  - added support for companion OME-TIFF filesets where TIFF does not link back to the metadata file
  - improved handling of missing planes in TiffData
- PerkinElmer Operetta
  - improved support to handle datasets generated by the Harmony software
- TIFF
  - split IFDs into separate series if the dimensions or pixel type mismatch
  - restricted use case for legacy TIFF JAI reader
  - fixed a bug with FillOrder which resulted in 0 pixel values
- Zeiss CZI
  - reduced duplicate original metadata when reading a pyramid file
- Zeiss TIFF
  - added support for AVI files acquired with Keyence software
- Zeiss ZVI
  - reuse stream for sequential calls to `openBytes` on the same plane
- updated all pyramidal format readers to consume `SubResolutionReader`
- updated all readers to consume `MetadataTools` getter to retrieve enumerations
- reviewed all readers and plugins to close open instances of `RandomAccessInputStream`

- fixed some deprecation warnings in a number of readers
- for RGB images using `ChannelSeparator` all channel metadata is now copied instead of just names

ImageJ plugin improvements:

- updated the updater message in the Fiji plugin (thanks to Jan Eglinger)
- disabled LUT writing for any plane that has a default grayscale lookup table
- added macro option to always skip LUT writing

MATLAB toolbox improvements:

- improved performance of bfGetPlane by removing an unnecessary data copy (thanks to Cris Luengo)

Command-line tools improvements:

- `bfconvert` utility
  - added `-no-flat` option to the command-line tools to convert files with sub-resolutions
  - added `-pyramid-scale` and `-pyramid-resolutions` options to generate sub-resolutions during conversion
  - removed `Plate` elements when `-series` is passed as an option
  - extended usage to describe available formats, extensions and compressions
- `xmlvalid` utility
  - added new `validate` methods to `loci.formats.tools.XMLValidate` returning the validation status
  - added a return code to `xmlvalid`

Component changes:

- *ome-common* was upgraded to 6.0.0
- *ome-codecs* was upgraded to 0.2.3
- *ome-model* was upgraded to 6.0.0

Automated test changes:

- added `testng.allow-missing` property allowing to skip unconfigured filesets
- added `testUnflattenedSaneOMEXML` to compare series count to OME-XML images count when resolution flattening is disabled
- added `test-equivalent` target to compare pixel data between two files
- added support for storing resolution index and resolution count in the configuration files used for automated testing
- tests now fail when a configured file throws UnknownFormatException

Documentation improvements:

- fixed the **`xmlvalid`** documentation page (thanks to Kouichi C. Nakamura)
- improved the memory section of the MATLAB documentation page (thanks to Kouichi C. Nakamura)
- extended `IFormatReader` Javadocs to reflect the reader guide
- added reference to current Adobe TIFF specification
- switched to image.sc as the reference location for public feedback

5.40 5.9.2 (2018 September 03)

File format fixes and improvements:

- **AVI**
  - added support for AVI files acquired with Keyence software
- **Gatan**
  - fixed a bug when reading a file with an empty tag of type 23
- **Deltavision**
  - extended the objective metadata support (thanks to David Pinto)
- **MRC**
  - fixed the reading of MRC files generated with FEI EPU software
- **Zeiss LSM**
  - improved the channel color detection for SIM data

Component changes:

All OME dependencies were upgraded mostly with build changes and documentation improvements:

- ome-common was upgraded from 5.3.2 to 5.3.6
- ome-poi was upgraded from 5.3.1 to 5.3.3
- ome-mdbtools was upgraded from 5.3.1 to 5.3.3
- ome-jai was upgraded from 0.1.0 to 0.1.3
- ome-codecs was upgraded from 0.2.0 to 0.2.2
- ome-stubs was upgraded from 5.3.0 to 5.3.2
- ome-model was upgraded from 5.5.4 to 5.6.3

Documentation improvements:

- added links to public sample files for Imaris IMS, DICOM, Leica-SCN, LEO, MRC, PNG, TIFF and Trestle formats

5.41 5.9.1 (2018 August 14)

File format fixes and improvements:

- **Olympus OIR**
  - fixed a bug to prevent incorrect files from being read when multiple datasets are in the same location
- **LEO**
  - updated parsing of metadata values for image pixel size, working distance, filament, EHT and date (thanks to David Mankus)
- **DeltaVision**
  - reader can now detect up to 12 channels
- **Micro-Manager**
  - now logs a warning when an image is acquired with an unsupported version

Documentation improvements:

- added QuPath to the list of visualization and analysis applications
- updated the link to the i3dcore library
- updated the link to Slidebook
- improved MATLAB documentation with information on Java heap memory preferences (thanks to Kouichi C. Nakamura)
- corrected a number of permanently redirected URLs in the component and format pages

5.42 5.9.0 (2018 July 3)

File format fixes and improvements:

- **MetaMorph**
  - fixed a `NullPointerException` when a stage label is not present
  - ensured that reported domain is now consistent with the existence of a Plate in OME-XML
  - fixed Metamorph RGB series channel count (thanks to Jeremy Muhlich)
- **Leica LIF**
  - improved handling of dimension order for non-RGB channels
- **Imspector OBF**
  - added support for FLIM datasets
- **Inveon**
  - updated to attempt to locate renamed data files
- **Volocity**
  - expanded image names to include the stack parent names
- **Olympus OIR**
  - added a fix for slow tag reading and a potential infinite loop
- **TIFF**
  - added support in `TiffWriter` for the writing of DEFLATE (zlib) compression
  - deprecated `getIFDs()` in `TiffParser` and added `getMainIFDs()` and `getSubIFDs()`
- **Zeiss CZI**
  - fixed an issue with big images when tiling is present but a pyramid is not
- **Nikon NIS-Elements ND2**
  - prevented integer overflow exception when reading a tile from a large image
- **Amersham Biosciences Gel**
  - prevented overflow issue when reading unsigned integer values
- **Cellomics**
  - fixed indexing when the field counts are variable
- **Trestle**

  - updated to ensure consistent ordering of used files

Bug fixes and improvements:

- enabled building and testing with Java 9 and 10
- added CI testing with Java 10 on AppVeyor and Travis
- removed Java 7 from Appveyor matrix
- updated a number of Maven plugins to current versions
- corrected warnings in Maven configuration in sub-components
- added a warning to clarify the behavior when passing metadata with `dimensionOrder` in `bfsave` as part of the Bio-Formats MATLAB toolbox (thanks to Jonathan Armond)
- improved robustness in the detection of patterns as part of the file stitching
- fixed a bug relating to dimension order in the Bio-Formats plugins Exporter
- fixed download URLs in Bio-Formats command-line tools
- updated use of `static final` to match Oracle's recommendations and convention
- disabled upgrade checker when running unit tests
- added support to data repo test suite for unconfigured tests

Documentation improvements:

- fixed unstable links flagged by automated link checking
- begun adding testing for breakages to memo files
- clarified ordering expectation in `getUsedFiles` Javadocs
- added documentation for dimensionOrder in bfsave with the *MATLAB toolbox*
- fixed broken links in previous release notes
- expanded documentation for command-line tools to cover undocumented *options* and *environment variables*
- added a new license/copyright section to the *About Bio-Formats* page
- updated the public format page for the Vectra QPTIFF format

5.43 5.8.2 (2018 April 23)

File format fixes and improvements:

- **JPEG**
  - large images with no restart markers now revert to using `DefaultJPEGReader` for improved decoding
- **Micro-Manager**
  - when available `PositionName` will be parsed and used as the image name
- **Hamamatsu ndpi**
  - updated image names to be more meaningful when resolutions are not flattened
- **InCell 2000/6000**
  - fixed an `IllegalArgumentException` and improved well and field indexing
- **AVI**

  - fixed a bug with padding for RGB images

- **NIfTI**
  - the `nDimensions` field is now used to read additional dimensions when size is greater than 4
- **PerkinElmer Opera Flex**
  - fixed a bug which resulted in an incorrect field count
- **Zeiss CZI**
  - improved handling of files with no extension

Bug fixes and improvements:

- an error message is now logged by `ImageReader` when finding a reader for an empty file
- added a new protected helper method to `Memoizer` to check if a directory is writable
- improved the rounding of `PlanePosition` values for data repo configuration testing
- prevented a null pointer exception when retrieving plane exposure time using Bio-Formats ImageJ macro extensions
- updated `MinMaxCalculator` to account for unflattened multi resolution images

Documentation improvements:

- decoupled the Bio-Formats documentation to the new ome/bio-formats-documentation GitHub repository
- updated *Adding format/reader documentation* for the new decoupled workflow
- improved link checking in automated builds

5.44 5.8.1 (2018 March 22)

File format fixes and improvements:

- **TIFF**
  - updated TiffWriter so that planes will no longer be split when using non-standard SamplesPerPixel e.g. images with 2 or 4 samples per pixel. This will ensure the `TiffData` elements represent the structure specified by the user. If users wish to split planes the `ChannelSeparator` and `bfconvert` provide the means to do this explicitly
  - updated TiffWriter to use the correct logic for index checking when writing tiled images
  - fixed a `ClassCastException` when the `NEW_SUBFILE_TYPE` tag has a non-standard type or count such that the value is not inlined
  - updated to also check the last IFD for an ImageJ comment in the scenario that the image has been processed by other software
- **NRRD (Nearly Raw Raster Data)**
  - added support for `space directions` and `space units` fields added in version 4
- **Evotec/PerkinElmer Opera Flex**
  - updated to read rather than calculate image offsets when a single tile is used

Bug fixes and improvements:

- limited the number of exceptions in the Bio-Formats plugins exporter when an unsupported pixel type is found

- fake test images now allow for per-plane ExposureTime{X,Y,Z} and Position{X,Y,Z} keys in the INI file (for further details see the documentation for *Generating test images*)
- file patterns now have expanded support for multi-channel pyramids, allowing for the matching of at least two channels rather than three, and the stitching of files containing a pyramid has also been fixed

Documentation improvements:

- improved testing of external links

5.45 5.8.0 (2018 February 21)

New file formats:

- **Ionpath MIBI**
  - added a new reader to support the reading of Ionpath Multiplexed Ion Beam Imaging (MIBI) files (thanks to Rachel Finck)
- **PerkinElmer Vectra QPTIFF**
  - added support for PerkinElmer Vectra QPTIFF files (The QPTIFF Bio-Formats reader is provided through a collaboration between PerkinElmer, Inc and Glencoe Software Inc.)

File format fixes and improvements:

- **cellSens VSI**
  - added support for lossless JPEG compression
- **Imspector OBF**
  - improved the parsing of OBF files with embedded OME-XML metadata (thanks to Bjoern Thiel)
- **Leica LIF**
  - companion metadata files are now attached if present
- **Micro-Manager**
  - fixed a bug related to the parsing of the metadata closing block
- **NRRD (Nearly Raw Raster Data)**
  - added support for GZIP pixel stream contained within a .nrrd file
- **Olympus OIR**
  - added support for multi-file datasets
- **OME-TIFF**
  - when files are ungrouped the dimensions are corrected by checking the indexes for each associated TiffData
- **PerkinElmer Operetta**
  - added support for additional metadata fields such as `Instrument`, `Wavelength` and `Exposure time`
- **TIFF**
  - fixed a bug when printing IFD values of type `OnDemandLongArray`
  - fixed a bug when writing tile sizes for multi-series images
- **Zeiss CZI**
  - when Z positions are not enumerated then values are calculated from a Z step

  - metadata for DisplaySetting will now be preserved in the original metadata table

Bug fixes and improvements:

- removed unused ScreenReader in preparation for migrating it to be an external reader
- fixed a bug with the generation of thumbnails in Bio-Formats plugins
- updated the Maven POM to unify component version property names
- tile size is now reported in the core metadata when using the showinf tool
- added `setFilePatternIds` to `ImporterOptions` for use with Bio-Formats plugins
- improved the precision of format identification for MRC, I2I, and Zeiss LSM

Documentation improvements:

- fixed and updated a number of external documentation links
- added links to public NRRD samples

5.46 5.7.3 (2018 January 11)

File format fixes and improvements:

- **TIFF**
  - fixed a NullPointerException when reading a TIFF file from the root system directory
  - improved support for large images that are stored as a single uncompressed tile with multiple interleaved channels
- **MRC (Medical Research Council)**
  - added support in original metadata for the fields `ISPG` and `Is data cube`
- **TillPhotonics TillVision**
  - directory listings for .pst files are now sorted
- **MetaMorph**
  - directory listings are now sorted during file initialization
- **Amira Mesh**
  - now supports `Avizo` in the file header in addition to the existing support for `AmiraMesh`
- **Becker & Hickl SPCImage**
  - added a fix for IllegalArgumentException when reading files with compressed data
- **Zeiss CZI**
  - fixed an IndexOutOfBoundsException when creating ROI objects

Bug fixes and improvements:

- removed unused target utils-formats-api from ant build
- automated Memoizer tests updated to use UUID for generating unique memo file directories
- detect and fix Findbugs' `SBSC_USE_STRINGBUFFER_CONCATENATION` using StringBuilder
- configuration files for the automated test suite now use raw physical size rather than formatted size
- added first version of Dockerfile for running the automated test suite standalone

Documentation improvements:

- added a support page to the Bio-Formats project
- updated reference URLs for the Aperio ImageScope and Micro-Manager
- documented issues with conflicts in the *JAI ImageIO component*
- clarified the default values of HCS keys for fake images in the documentation for *Generating test images*
- corrected external links which failed automatic link checking

5.47 5.7.2 (2017 November 21)

File format fixes and improvements:

- Nikon ND2 - fixed a bug which would use the incorrect channel count for small-sized single channel images
- **MetaMorph TIFF**
  - changed the reader's behaviour to populate exposure times for all planes when only a single exposure time is defined
- **DeltaVision**
  - improved parsing of the associated log files to add additional key value pairs to global metadata
- **EPS (Encapsulated PostScript)**
  - fixed an exception when reading pixel data in cases with embedded TIFF
- **GIF**
  - fixed a bug to display the correct data when reading planes out of order

Bug fixes and improvements:

- fixed failures with Ant build from a clean Maven repository by updating Maven repositories to use HTTPS rather than HTTP
- now using safe version checking for Bio-Formats plugins to prevent a bug with Java 9
- updated the JPEG-XR codec to allow either interleaved or non-interleaved data to be returned

Documentation improvements:

- added clarification regarding Bio-Formats version requirements for using Java 7 or above
- updated download links to latest Bio-Formats release version
- updated the link to the most active fork of JAI ImageIO
- fixed a number of external broken links
- added a Trello link for contributing external developers
- added a link to the page *Adding format/reader documentation pages* to help those contributing to the documentation or supported formats pages
- the *Bio-Rad Gel* page has been updated to add a link to biorad1sc_reader, an external python implementation (thanks to Matthew Clapp)

5.48 5.7.1 (2017 September 20)

File format fixes and improvements:

- **Nikon NIS-Elements ND2**
  - improved parsing of Z position values
- **LaVision Imspector**
  - corrected the value of time per FLIM channel
  - fixed a bug which saw the Z and T dimensions swapped
  - fixed a divide by zero exception
  - added a fix for incorrect time-base and number of channels
- **TIFF**
  - added support for handling files with a FillOrder of 2 in which the bits in each byte are reversed
  - improved support for multi-channel ImageJ TIFF files greater than 4GB in size

Performance improvements:

- improved TIFF performance by using non-regexp String replacement (thanks to Thushara Wijeratna)
- improved TIFF handling of Strings for large metadata (thanks to T. Alexander Popiel)

Documentation improvements:

- updated documentation to reference support for ImageJ TIFFs
- added links to format options page to user and developer index pages

5.49 5.7.0 (2017 September 4)

File format fixes and improvements:

- **Imaris HDF**
  - fixed resolution problems in which dimensions and resolution order were incorrectly calculated (thanks to Eliana Andreica)
- **Nikon NIS-Elements ND2**
  - fixed a bug in offset calculation when native chunk map is being used
- **MetaMorph**
  - corrected delta T and position Z values for multi-channel images when channels are split across multiple files
- **Amnis FlowSight**
  - better handling of exceptions in isThisType method (thanks to Claire McQuin)
- **PicoQuant Bin**
  - better handling of exceptions in isThisType method (thanks to Claire McQuin)

Bug fixes and improvements:

- reviewed and corrected URLs throughout the Bio-Formats source code
- updated Bio-Formats Macro Extensions list with a missing function

- added a new option in Bio-Formats plugins to configure the slice label display using patterns

Documentation improvements:

- added new format page for *OMERO Pyramid*
- updated the developer page for *Working with whole slide images*
- added new page for configuring options in *Bio-Formats plugins*
- updated documentation sidebar to enable navigation of previous versions

5.50 5.6.0 (2017 August 14)

File format fixes and improvements:

- **Zeiss CZI**
  - added support for images from Elyra PALM system
  - prevented a potential infinite loop when a scene with a pyramid is missing
- **cellSens VSI**
  - a new option has been added to throw an exception rather than logging a warning if .ets file is missing. The option, `cellsens.fail_on_missing_ets`, can be used via the MetadataOptions API, as a parameter in the command line tools or via the Bio-Formats configuration dialog in ImageJ
- **MetaMorph Stack (STK)**
  - fixed an error with HCS style datasets always returning the first plane regardless of the requested index
  - updated to use stage labels starting with `Scan` to detect when a whole plate is saved in a single .stk file
  - fixed a bug for `ArrayIndexOutOfBoundsException` when an image contains a single Z plane
- **Gatan Digital Micrograph**
  - added support for Z stacks and ROIs
  - fixed several bugs in tag parsing
- **PerkinElmer Operetta**
  - ensure TIFF files exist before reading
- **JPEG**
  - support added for images with more than `Integer.MAX_VALUE` pixels

Bug fixes and improvements:

- **JPEGTileDecoder**
  - class now implements AutoCloseable to prevent resource leaks
- **Bio-Formats Plugin**
  - improved performance when using options to concatenate multiple series together
- **TiffSaver**
  - made performance improvements to prevent the writing of a new IFD for each tile, resulting in significant file size reductions for images with a large quantity of tiles

Documentation improvements:

- updated website and URL links for new OME Website website

- added missing *Andor SIF* to supported formats page
- added a new page *Working with whole slide images* outlining the API support for pyramids/resolutions
- fixed broken documentation links for external resources which are no longer available
- updated the style of Sphinx documentation

Component architecture changes/decoupling:

- decoupled image encoding and decoding routines to the new ome/ome-codecs GitHub repository and consumed as 'org.openmicroscopy:ome-codecs' artifact from Maven Central
- removed components/forks/jai - decoupled to the new ome/ome-jai GitHub repository and consumed as part of 'org.openmicroscopy:ome-jai' artifact from Maven Central
- replaced components/formats-api/codecs classes with wrappers around 'org.openmicroscopy:ome-codecs'
- replaced components/formats-bsd/codecs classes with wrappers around 'org.openmicroscopy:ome-codecs'

Updated build system:

- ant now removes the build files of the bundles during 'clean' to prevent a mix of dependencies

5.51 5.5.3 (2017 July 5)

File format fixes and improvements:

- **Zeiss CZI**
  - fix to store Bézier ROIs as polygons, using the control points for the set of Bézier curves to form an approximation of the ROI
  - improved parsing of stage positions in metadata
  - improved parsing of detector gain values
  - removed OME-XML validation errors by fixing potential for duplicate detector IDs
  - removed invalid XML failures for Modulo label elements
  - time increment metadata now populated on `Pixels` element
  - fix to deal with consecutive empty planes in a series (thanks to Nicholas Trahearn)
- **DICOM**
  - no longer allow core metadata to be modified when determining if files belong to a DICOM dataset
- **Nikon NIS-Elements ND2**
  - fixed calculation for scanline padding
- **Kodak BIP**
  - stricter file type checking enforced by no longer relying only on the file suffix
- **MINC MRI**
  - improved parsing of metadata by correcting units for physical sizes, pixel type and capturing XYZ plane positions in OME-XML
- **Bio-Rad Gel**
  - fixed the width of pixel data offset field
- **DeltaVision**

  - improved accuracy of format detection checking for input streams

- **Andor SIF**
  - fixed support for cropped images by parsing bounding box of the stored image

Documentation improvements:

- Olympus cellSens VSI updated to include list of available specifications

5.52 5.5.2 (2017 June 15)

File format fixes and improvements:

- **Olympus FluoView FV1000**
  - fix for `java.lang.ArrayIndexOutOfBoundsException` caused by filter names of "—" (thanks to Stefan Helfrich)
  - refactored channel metadata population and increased usage of `DataTools` utility functions
- **Zeiss CZI**
  - fixed detection of Z line scans that caused incorrect dimensions in certain filesets
  - improved exception handling of truncated/invalid files
- **Veeco AFM**
  - fixed reading of tiled images
- **Hamamatsu ndpi**
  - prevented potential memory leak by ensuring all `TiffParser` streams are closed

Bug fixes:

- **OMEXMLServiceImpl**
  - improved exception handling to deal with potential `java.lang.NullPointerException` when unable to locate OME-XML version while attempting to transform to the latest version

Documentation improvements:

- updated documentation to be compatible with the latest version of Sphinx 1.6
- fixed the usage/references of the option markup in documentation
- fixed the table in the Micro-Manager user page
- updated metadata ratings for supported formats

Updated build system:

- **OME-Model version bump**
  - the ome-model component has been updated to 5.5.4 which includes improvements to performance, documentation and the C++ model implementation

5.53 5.5.1 (2017 May 25)

File format improvements:

- **CellH5**
  - fix for `HDF5SymbolTableException` when recycling an IFormatReader to reopen another CellH5 file

  - bug fix related to opening of subsets of CellH5 files, namely `openBytes(r, no, x, y, w, h)` for y>0

- **Zeiss CZI**
  - fix pyramid resolution indexing for pyramids of different depths
  - fix for incorrect channel names and colors
- **Zeiss AxioVision ZVI**
  - correct parsing of epoch for Zeiss TIFF and Zeiss ZVI

Bug fixes:

- **Command line tools**
  - fix for `java.lang. NegativeArraySizeException` caused by incorrect dimensions when using showinf via command line with options set to autoscale and crop
- **Format tools**
  - fix for `java.lang. IndexOutOfBoundsException` when using `getFilename` with an image containing multiple samples per pixel channels and a single effective channel

Updated build system:

- **Autogen jobs**
  - fix for `gen-meta-support` to locate available `org.openmicroscopy:ome-xml` sources from the Maven repository following the decoupling of the model components
- **FileHandleTest**
  - exclude JHDF5 native libraries from `FileHandleTest` to enable CellH5 files to be included in daily tests

Documentation improvements:

- added a new example file for reading and writing of XZ and YZ orthogonal planes

5.54 5.5.0 (2017 May 8)

New file formats:

- **Olympus OIR**
  - added support for *Olympus .oir* data (funded by a partnership between Glencoe Software and OLYMPUS EUROPA SE & Co. KG)
- **PerkinElmer Columbus**
  - added support for *PerkinElmer Columbus* data

File format improvements:

- **Andor Bio-Imaging Division (ABD) TIFF**
  - fixed acquisition date format from `MM/dd/yyyy` to `dd/MM/yyyy`
- **Nikon NIS-Elements ND2**
  - corrected logic used to determined `PixelType` by parsing uiBpc tags
- **Hamamatsu ndpi**
  - improved handling of channels in NDPIS datasets (thanks to Manuel Stritt)

- **Imspector OBF**
  - fix for `SAXParseException` when description field in metadata is empty

Documentation improvements:

- added links to public sample files for Cellomics
- added links to public sample files for InCell 3000

5.55 5.4.1 (2017 April 13)

File format improvements:

- **MIAS (Maia Scientific)**
  - added a fix for a possible exception when image files are not found under channel-specific subdirectories
- **BD Pathway**
  - added fix to check if `Experiment.exp` is a directory or an experiment file
- **Imspector OBF**
  - enabled forward compatibility for future versions, as the OBF format is backwards compatible (thanks to Bjoern Thiel)

Documentation improvements:

- updated external homepage link for FocalPoint
- removed Imago from list of visualization and analysis applications as it is no longer available from the Mayachitra website
- added links to public sample files for Hamamatsu NDPI and Hamamatsu VMS
- listed OpenSlide as available software for supported formats
- added a new developer page detailing in-memory reading and writing
- updated the Bio-Formats API versioning policy, which now follows strict semantic versioning
- a new options page has been added, detailing the usage of configurable format-specific options for readers and writers. Links to the available options are also included under the relevant supported formats

5.56 5.4.0 (2017 March 21)

File format improvements:

- **DICOM**
  - added support for DICOMDIR files, which allow multiple DICOM files in a single directory to be opened as a single dataset
  - plane position values for values X, Y and Z are now being set in OME-XML
  - correctly read the physical size X and Y values based on the available specification
- **Nikon NIS-Elements ND2**
  - performance improvements based on reading chunkmap. Processing of the chunkmap can be disabled via the MetadataOptions API using the boolean option `nativend2.chunkmap`. For ImageJ users this option can be accessed via a checkbox in the Nikon ND2 section of the Bio-Formats configuration dialog *Plugins* → *Bio-Formats* → *Bio-Formats Plugins Configuration* (thanks to Christian Sachs)

- **OME-TIFF**
  - added an option to save an OME-TIFF dataset as a binary TIFF and companion XML. This can be used via the bfconvert command line tool by setting the value of option `ometiff.companion` to the name of the companion file to use. For example `bfconvert -option ometiff.companion outputFile.companion.ome inputFile.tiff outputFile.ome.tiff`
- **CellVoyager**
  - metadata fixes specifically the naming of plates. Additional refactoring of the reader for general maintainability
- **Gatan Digital Micrograph**
  - previously missing Image-Instrument reference has been added to OME-XML
- **TiffSaver**
  - ensure open resources are closed under all possible scenarios
- **Zeiss CZI**
  - improved performance of large uncompressed images. When tiles from a large uncompressed image with no internal tiling are requested, only the specific tile specified in the call to `openBytes` is read from disk, instead of the entire image being read and then copied
- **Zeiss AxioVision ZVI (Zeiss Vision Image)**
  - ensure that the `bitsPerPixel` field is always set to match the final pixel type, and populate any channel colors that were parsed in the metadata. The bits per pixel update should only affect `uint16` or `int16` files where the acquisition bit depth is not a multiple of 8, and the RGB channel count is greater than 1

Updated build system:

- updated dependency for NetCDF to 4.3.22
- updated copyright headers from 2016 to 2017 and reviewed and fixed any incorrect header descriptions
- documentation has been migrated to use `.rst` file format for Sphinx files
- reviewed and cleaned up warnings such as unused variables and imports
- added CellVoyager datasets to automated testing via continuous integration
- unified the semantics for creating temporary directories within unit tests

Documentation improvements:

- fixed link for PerkinElmer UltraVIEW system
- fixed links for NIfTI public specification and data sets
- available software for Hamamatsu ndpi has been updated from NDP.view to NDP.view2

5.57 5.3.4 (2017 February 21)

Bug fixes:

- **ImageJ**
  - fix for a NullPointerException when exporting images that were not opened via the Bio-Formats importer, and thus do not have a complete OMEXMLMetadata store
- **Java 1.9**
  - fix compile and runtime errors to enable building with Java 1.9

- **ECAT7**
  - update to add support for different versions of ECAT7 files (thanks to Torsten Stöter)

Updated build system:

- updated dependency for ome-model in the POM to version 5.4.0. This allows for improved ROI handling by enabling support for Shape objects with Transform attributes. OME-XML schema version remains unchanged as OME schema 2016-06

Documentation improvements:

- new public sample files added for ECAT7 (thanks to Torsten Stöter)
- new public sample files added for Leica LIF (thanks to Michael Goelzer)
- new specification document (Version 3.2) for Leica LIF
- updated links to OMERO documentation as a result of decoupling

5.58 5.3.3 (2017 February 2)

Bug fixes:

- **ImageJ**
  - fix for issue when exporting from an ImagePlus that represents signed data. The pixel type will now remain unchanged as will the pixel values which had previously been scaled incorrectly
- **Command line tools**
  - fix for `java.lang.IllegalArgumentException` when using bfconvert via command line with option set to only convert a single time-point, channel or Z section
- **Tiff writing**
  - using TiffWriter to write tiled images now supports the writing of BigTIFF datasets

File format fixes:

- **Applied Precision CellWorX**
  - fix to now display the correct plate name and dimensions
- **NIFTI**
  - a few fixes for problems with byte alignment when reading non-core metadata from NIFTI headers
- **Leica LIF**
  - added support for timestamps of LIF files created with LAS AF 3.1 or newer. In the case of a halted acquisition only non-null timestamps are stored in the OME metadata (thanks to Michael Goelzer)
  - the physical pixel height and width were incorrectly calculated by dividing by the number of pixels. This has now been corrected to match the official Leica LIF specification documents by dividing by the number of pixels minus one (thanks to Michael Goelzer)
  - for backwards compatibility an option to preserve pre-5.3.3 physical sizes has been added. This can be set either via command line tools, through the API with the `loci.formats.in.DynamicMetadataOptions` class, or in the Bio-Formats plugin configuration in ImageJ
- **Improvision TIFF**
  - channel colors are now being read and if present set correctly in image metadata
- **MetaMorph**

  - fix for `java.lang.OutOfMemoryError` exceptions when reading large Metamorph TIFF plates

Updated build system:

- version history file added to MATLAB bundle as NEWS.rst
- increased TiffWriter test coverage
- added test coverage framework for command line tools including new ImageConverterTest

Documentation improvements:

- improved documentation and new examples for using tiled writing
- updated developer documentation for use of Bio-Formats as a Maven, Gradle or Ivy dependency
- documentation for Leica LIF bug fixes and use of backward compatibility options
- fixes for a number of broken links

5.59 5.3.2 (2017 January 9)

Bug fixes:

- **ImageJ**
  - fixed race condition when opening multiple series from a dataset, as introduced by thumbnail loading changes in 5.3.0
  - updated thumbnail generation to be faster for datasets containing an image pyramid
- **Metamorph**
  - updated to read the refractive index and set `RefractiveIndex` on `ObjectiveSettings` in the generated OME-XML (thanks to Marc Bruce)
- **Metamorph TIFF**
  - fixed Z and channel dimension counts when each channel has a unique Z position
  - updated to read the emission wavelength and set `EmissionWavelength` on `LightSourceSettings` in the generated OME-XML
- **QuickTime**
  - fixed incorrect image data when reading of tiles from single channel files
- **file grouping**
  - fixed handling of `loci.formats.in.MetadataOptions` objects by the `loci.formats.FileStitcher` reader

Documentation improvements:

- fixed extensions listed for Zeiss TIFF
- simplified markdown for creating tables

5.60 5.3.1 (2016 December 19)

File format fixes:

- **TIFF**

  - fixed invalid IFD values when writing TIFF or OME-TIFF files with Bio-Formats 5.3.0. This bug affected the writing of TIFF and OME-TIFF via client code using `loci.formats.TiffWriter`, converting to a TIFF or OME-TIFF using 'bfconvert' command line tool or exporting to TIFF or OME-TIFF using ImageJ/FIJI Bio-Formats exporter.

5.61 5.3.0 (2016 December 12)

New features/API:

- added support for JPEG-XR compressed CZI data (funded by a partnership between Glencoe Software and ZEISS), adding 'ome:jxrlib' as a new dependency of Bio-Formats
- **improved tile-based image writing**
  - added new methods to the `loci.formats.IFormatWriter` interface allowing to set and retrieve the tile along the X and Y dimensions
  - added default implementations to the `loci.formats.FormatWriter` abstract class defaulting to the entire image width/height
  - added functionality to `loci.formats.TiffWriter` adding support for tiled images writing for TIFF and derived formats like OME-TIFF
  - added developer documentation and samples for tiled reading/writing
- added a new `MetadataOptions` implementation supporting arbitrary key/value pairs
- updated the display command line utility to support passing key/value options using *`showinf -option`*
- added two options to the CZI reader to disable autostitching and exclude pyramid file attachments. Added new checkboxes to the CZI configuration interface of the ImageJ plugin to activate these options

Bug fixes/deprecations:

- deprecated `loci.formats.meta.MetadataConverter` in favor of `ome.xml.meta.MetadataConverter`
- updated method deprecated in Octave 4.2.0 (thanks to Carnë Draug)
- **OME-XML**
  - fixed handling of Mask BinData elements

Component architecture changes/decoupling:

- removed formats-common component - now decoupled to the new ome/ome-common-java GitHub repository and consumed as 'org.openmicroscopy:ome-common' artifact from Maven Central
- removed ome-poi component - now decoupled to the new ome/ome-poi GitHub repository and consumed as 'org.openmicroscopy:ome-poi' artifact from Maven Central
- removed specification, xsd-fu and ome-xml components - now decoupled to the new ome/ome-model GitHub repository and consumed as 'org.openmicroscopy:{specification,ome-xml}' artifacts from Maven Central
- removed mdbtools component - now decoupled to the new ome/ome-mdbtools GitHub repository and consumed as 'org.openmicroscopy:ome-mdbtools' artifact from Maven Central
- removed stubs components - now decoupled to the new ome/ome-stubs GitHub repository and consumed as 'org.openmicroscopy:{lwf-stubs,mipav-stubs}' artifacts from Maven Central
- removed metakit component - now decoupled to the new ome/ome-metakit GitHub repository and consumed as 'org.openmicroscopy:metakit' artifacts from Maven Central
- updated developer documentation for the decoupled components

Updated build system:

- dropped embedded JARs and now use the Maven Ant Tasks plugin to unify the dependencies using the POM
- improved Ant JAR and bundle target
- dropped deprecated osgi targets, OME Tools bundle and ome-jxr component
- removed PDF generation from the docs-sphinx target
- added version number to Javadoc zip bundle name
- migrated unit tests out of test-suite into formats-bsd
- fixed test-suite targets, paths and symlink handling
- fixed test-metadata and migrated it into test-suite
- fixed mismatch between `ND2HandlerTest` package and location
- cleaned up test-build to remove obsolete and decoupled components and folders
- simplified Travis build
- POM repositories clean-up to reduce complexity and use Maven Central as the first location to look for dependencies
- now storing all versions in the top-level POM
- updated build versioning from Maven by unified versioning strategy, reviewing meta information stored in the manifests of each JAR and introspecting this information in the `FormatTools` API to retrieve version and revision numbers
- updated developer documentation on updated build system

5.62 5.2.4 (2016 October 18)

Java bug fixes:

- **OME-TIFF**
  - fixed regression when populating plane metadata
- **CZI**
  - populated series metadata with the scene/position information

5.63 5.2.3 (2016 October 5)

Java bug fixes:

- **CZI**
  - fixed imageCount for RGB images
- **ICS writing**
  - fixed ordering of image dimensions
- **DeltaVision**
  - fixed reading of large time dimensions

Command-line tools improvements:

- `bftools.zip` now includes the version history as `NEWS.rst` (thanks to Gerhard Burger)

Code clean-up/improvements:

- switched to *String.indexOf(int)* in GPL-licensed reader code so that a simpler library method can be used
- strings now extended with characters where possible
- completed deprecation of *DataTools.sanitizeDouble()*
- deprecated unused OSGi and ome-tools bundle build targets

OME-XML changes/improvements:

- bumped schema version number to 2 (schema namespace left unchanged)
- added acquisition modes *BrightField*, *SweptFieldConfocal* and *SPIM*
- added parsing for Laser Scan Confocal and Swept Field Confocal

Documentation improvements:

- documented versioning policy
- clarified supported versions for Micro-Manager and Olympus ScanR files

5.64 5.2.2 (2016 September 13)

Java bug fixes and improvements:

- fixed a regression in which the DataTools number parsing API would not be thread-safe anymore
- **InCell**
  - improved handling of Analyzer 2000 datasets to find TIFF files
- **FV1000**
  - fixed preview names ordering
- **OME-TIFF**
  - enabled all BigTIFF extensions
- various code cleanup across the Java code
- added test coverage for all example codes in the developer documentations
- added tests covering the semantics of the INI parser

ImageJ bug fixes and improvements:

- fixed a bug in ImageJ when swapping dimensions of an image with multiple series of different dimensions
- added an option to the exporter to pad filename indexes with zeros

Command-line tools improvements:

- allowed the binaries to be symlinked (thanks to Gerhard Burger)
- added an option to bfconvert to pad filename indexes with zeros

5.65 5.2.1 (2016 August 25)

Java bug fixes:

- **Zeiss CZI**
  - fixed NumberFormatException when the position object is not null but the values of child are null
- **SimplePCI**

    - made IniParser less stringent to allow reading of imperfectly formatted TIFF description headers
- fixed stitching of file patterns in ImageJ to remove duplication of directory names in the file path
- added an option to bfconvert to allow creation of OME-TIFF without lookup tables
- addition of MetadataOnly elements containing no BinData or TiffData now handled via MetadataTools API in ImageInfo
- example code in developer docs is now tested via a new Maven module

5.66 5.2.0 (2016 August 18)

Java format support improvements are listed below.

†Denotes a major breaking change to the reader (typically modification of core metadata). Code changes or re-import may be necessary in ImageJ/FIJI and OMERO.

- added support (and public sample files) for *Becker & Hickl .spc FIFO* data
- added support for *Princeton Instruments .spe* data
- **bug fixes for many formats including:**
    - **CellSens VSI†**
        * fixes for correctly reading dimensions
    - **FlowSight**
        * fixes to infer channel count from channel names (thanks to Lee Kamentsky)
    - **Hamamatsu VMS†**
        * fixed dimensions of full-resolution images
    - **ICS writing**
        * fixed dimension population for split files
    - **Kodak BIP**
        * fixed handling of CCD temperature stored in hexadecimal
    - **Leica LIF**
        * fixed incorrect plane offsets for large multi-tile files
    - **LiFlim**
        * fixed `ExposureTime` check and units usage
    - **Micro-Manager**
        * fixed handling of large datasets saved as image stacks and split over multiple files
        * added user documentation for file saving options
    - **MRC and Spider**
        * fixed format type checking
    - **Nifti**
        * fixed `planeSize` to prevent crashes when loading large files (thanks to Christian Niedworok)
        * added support for gzipped compressed .nii.gz files (thanks to Eric Barnhill)
        * added public samples and updated documented supported file extensions

- **OME-TIFF**
  - fixed `Plane` population errors
  - fixed `NullPointerException` when closing reader for partial multi-file filesets
  - reduced buffer size for `RandomAccessInputStreams` to improve performance
  - deprecated `getMetadataStoreForConversion` and `getMetadataStoreForDisplay` methods
- **OME-XML**
  - fixed metadata store
- **PicoQuant**
  - updated reader to always buffer data
- PNG writing
- **SDT**
  - performance improvements for loading of large files
- **Slidebook**
  - Slidebook6Reader is now completely external and fully maintained by 3i (see http://www.openmicroscopy.org/info/slidebook) and is specified as such in the `readers.txt` configuration file
- **SVS**
  - fixed `NumberFormatException`
- **Tiff**
  - fixed integer overflow to read resolutions correctly
  - fixed handling of tiled images with tile width less than 64
- **Zeiss CZI**
  - fixed timestamp indexing when multiple separate channels are present
  - improved slide support - slides are now detected as a complete full-resolution image (instead of each tile being a separate series) and pyramid sub-resolutions and label/overview images are also detected
- **Zeiss LSM**
  - fixed `Plane` population errors
- **Zeiss ZVI†**
  - reworked image ordering calculation to allow for tiles

Top-level Bio-Formats API changes:

- Java 1.7 is now the minimum supported version
- the native-lib-loader dependency has been bumped to version 2.1.4
- the xalan dependency has been bumped to version 2.7.2
- all the ome.jxr classes have been deprecated to make clear that there is no JPEG-XR support implemented in Bio-Formats as yet
- **the DataTools API has been extended to add a number of utility functions to:**

  - account for decimal separators in different locales
  - parse a `String` into `Double`, `Float`, `Integer` etc
  - handle `NumberFormatException` thrown when parsing Unit tests
- the Logging API has been updated to respect logging frameworks (log4j/logback) initialized via a binding-specific configuration file and to prevent `DebugTools.enableLogging(String)` from overriding initialized logger levels (see *Logging* for more information)
- helper methods have been added to FormatTools allowing a stage position to be formatted from an input `Double` and an input unit
- the Formats API has also been updated to add a new validate property to `MetadataOptions` and support for `MetadataOptions` has been moved to FormatHandler level to allow it to be used by both Readers and Writers
- initial work on Reader discoverability extended the ClassList API to allow the `readers.txt` configuration file to be annotated using key/value pairs to mark optional Readers and specify additional per-Reader options

Other general improvements include:

- improved performance of `getUsedFiles`
- fixes for `FilePatternBlock`, `AxisGuesser`, `FilePattern`
- fixes for the detection of CSV pattern blocks by `FilePatternBlock`
- `bioformats_package.jar` now includes bio-formats-tools as a dependency so `ImageConverter`, `ImageFaker` and `ImageInfo` classes are included in the bundle
- the JACE C++ implementation has been decoupled as it does not function with Java 1.8 (see legacy repo)
- **ImageJ fixes**
  - to allow reader delegation when a legacy reader is enabled but not working
  - to allow ROIs to be imported to the ImageJ ROI manager or added to a new overlay
- **MATLAB fixes**
  - improved integration with Octave (thanks to Carnë Draug)
  - added logging initialization
- **Command-line tools fixes**
  - upgrade check no longer run when passing -version
  - common methods refactoring
  - showinf improvements to preload format
  - tiffcomment now warns that it requires an ImageDescription tag to be present in the TIFF file
- added many automated tests and improved FakeReader testing framework
- **documentation improvements include:**
  - clarifying status of legacy Quicktime and ND2 readers
  - noting that the Gatan reader does not currently support stacks
  - more Java examples added to the developer documentation
  - new units page for developers

The Data Model version 2016-06 has been released to introduce Folders, and to simplify both the graphical aspects of the model and code generation. Full details are available in the OME Model and Formats Documentation. OME-XML changes include:

- *Map* is now a complexType rather than an element and *MapPairs* has been dropped
- extended enum metadata has been introduced to better support units
- *Shape* and *LightSource* are now complexTypes rather than elements
- BinData has been added to code generation to handle raw binary data
- **various code generation improvements to:**
  - simplify and standardize the generation process
  - remove a number of hard-coded exceptional cases allowing for easier maintenance and growth
  - allow for genuine abstract model types and enable C++ model implementation
- updated OME-XML and OME-TIFF public sample files

The Bio-Formats C++ native implementation has been decoupled from the Java codebase and will be released as OME-Files C++ from now on, with the exception of OME-XML which is still within Bio-Formats at present (there is a plan to decouple both the Java and the C++ versions of OME-XML in future).

The following components have had their licensing updated to Simplified (2-clause) BSD:

- XSL transforms
- specification code
- xsd-fu Python code

5.67 5.1.10 (2016 May 9)

Java bug fixes:

- fixed warnings being thrown for ImageJ and other non-FIJI users on Windows (these warnings were triggered by the removal of the 3i Slidebook DLLs from the source code repository in Bio-Formats 5.1.9 and should now only be triggered when opening Slidebook files without the update site enabled - http://www.openmicroscopy.org/info/slidebook)
- a fix in the ImageJ plugin for files grouped using the "Dimensions" option
- a fix for writing TIFF files in tiles

5.68 5.1.9 (2016 April 14)

- **Java bug fixes, including:**
  - **SDT**
    - fixed width padding calculation for single-pixel image
  - **Deltavision**
    - fixed the parsing of the new date format
    - added support for parsing and storing the working distance in native units
  - **Micromanager**
    - cleaned up JSON metadata parsing
  - **Olympus Fluoview**
    - fixed null pointer exceptions while parsing metadata

  - **Leica LIF**
    - fixed large multi-tiled files from having incorrect plane offsets after the 2GB mark
  - **EM formats (MRC and Spider)**
    - added native length support for EM readers
  - **Gatan**
    - fixed erroneous metadata parsing
    - added support for parsing and storing the physical sizes in native units
  - **OME-TIFF**
    - improved handling of OME-TIFF multi-file fileset's with partial metadata blocks
  - **Nikon ND2**
    - fixed the parsing of emission wavelength
  - **Olympus CellR (APL)**
    - fixed multiple parsing issues with the mtb file
  - **SlideBook**
    - removed slidebook dlls from Bio-Formats repository
    - http://www.openmicroscopy.org/info/slidebook
  - **Zeiss CZI**
    - fixed parsing of files with multiple mosaics and positions
- **Documentation updates, including:**
  - improved documentation for the export of BigTIFFs in ImageJ
- **C++:**
  - no changes.

5.69 5.1.8 (2016 February 15)

- **Java bug fixes, including:**
  - **FEI TIFF**
    - fixed stage position parsing and whitespace handling (thanks to Antoine Vandecreme)
  - **Pyramid TIFF**
    - fixed tile reading when a cache (.bfmemo) file is present
  - **MicroManager**
    - updated to parse JSON data from tags 50839 and 51123
    - fixed to detect `*_metadata.txt` files in addition to `metadata.txt` files
    - fixed to handle datasets with each stack in a single file
  - **OME-XML**
    - updated to make .ome.xml an official extension
  - **OME-TIFF**

        * fixed to ignore invalid BinaryOnly elements
    - **TIFF**
        * fixed caching of BigTIFF files
    - **Slidebook**
        * fixed handling of montages in Slidebook6Reader (thanks to Richard Myers)
    - Performance improvement for writing files to disk (thanks to Stephane Dallongeville)
    - **Build system**
        * fixed Maven POMs to reduce calls to artifacts.openmicroscopy.org
        * fixed bioformats_package.jar to include the loci.formats.tools package
- **Documentation updates, including:**
    - updated format pages to include links to example data
    - clarified description of Qu for MATLAB (thanks to Carnë Draug)
    - added installation instructions for Octave (thanks to Carnë Draug)
- **C++:**
    - Bugfixes to the OME-TIFF writer to correct use of the metadata store with multiple series
    - Ensure file and writer state consistency upon close failure

5.70 5.1.7 (2015 December 7)

- **Java bug fixes, including:**
    - Prevent physical pixel sizes from being rounded to 0, for all formats
    - **Metamorph**
        * fixed calculation of Z step size
        * fixed detection of post-processed dual camera acquisitions (thanks to Mark Kittisopikul)
    - **OME-XML**
        * fixed XML validation when an 'xmlns' value is not present (thanks to Bjoern Thiel)
    - **MINC**
        * fixed endianness of image data
    - **Andor/Fluoview TIFF**
        * fixed calculation of Z step size
    - **MATLAB**
        * improved performance by reducing static classpath checks (thanks to Mark Kittisopikul)
    - **Gatan**
        * fixed physical size parsing in non-English locales
    - **Automated testing**
        * fixed handling of non-default physical size and plane position units
- **Documentation updates, including:**

  - updated MapAnnotation example to show linkage of annotations to images

- **C++:**
  - no changes, released to keep version numbers in sync with Bio-Formats Java

5.71 5.1.6 (2015 November 16)

- **Java bug fixes, including:**
  - **Updated to use native units for following formats:**
    - IMOD
    - Analyze
    - Unisoku
    - Olympus CellR (APL)
  - **Metamorph TIFF**
    - fixed handling of multi-line descriptions
    - added support for dual camera acquisitions
  - **Zeiss LMS**
    - fixed exception in type detection
  - **Zeiss CZI**
    - fixed detection of line scan Airyscan data
  - **Slidebook**
    - fixed calculation of physical Z size
  - **ImageJ plugins**
    - fixed handling of non-default units
    - fixed setting of preferences via macros
  - **Automated testing**
    - fixed handling of non-default units for physical sizes and timings
- **C++ changes, including:**
  - allow relocatable installation on Windows
  - reduce time required for debug builds
- **Documentation updates, including:**
  - addition of "Multiple Images" column to the supported formats table
  - addition of a MapAnnotation example

5.72 5.1.5 (2015 October 12)

- **Java bug fixes, including:**
  - **ImageJ plugins**
    - fixed use of "Group files…" and "Open files individually" options

        * fixed placement of ROIs
        * fixed size of the "About Plugins > Bio-Formats Plugins" window
    - **xsd-fu (code generation)**
        * removed OMERO-specific logic
    - **Metamorph**
        * fixed physical Z size calculation
    - **Gatan DM3/DM4**
        * fixed physical pixel size parsing
    - **BMP**
        * added support for RLE compression
    - **DICOM**
        * updated to respect the WINDOW_CENTER tag
        * fixed image dimensions when multiple sets of width and height values are present
    - **Fluoview and Andor TIFF**
        * fixed physical Z size calculation
    - **Imspector OBF**
        * updated to parse OME-XML metadata (thanks to Bjoern Thiel)
- **C++ changes:**
    - TIFF strip/tile row and column calulations corrected to compute the correct row and column count
    - Several compiler warnings removed (false positive warnings in third-party headers disabled, and additional warnings fixed)
    - It is now possible to build with Boost 1.59 and compile with a C++14 compiler
- The source release is now provided in both tar.xz and zip formats
- **Documentation updates, including:**
    - **substantial updates to the format pages**
        * improved linking of reader/writer classes to each format page
        * improved supported metadata pages for each format
        * updated format page formatting for clarity
        * added developer documentation for adding and modifying format pages

5.73 5.1.4 (2015 September 7)

- **Bug fixes, including:**
    - **Command line tools**
        * fixed display of usage information
    - **Automated testing**
        * fixed problems with symlinked data on Windows

    * added unit tests for checking physical pixel size creation

- **Cellomics**
    * fixed reading of sparse plates
- **SlideBook**
    * fixed a few lingering issues with native library packaging
- **SimplePCI/HCImage TIFF**
    * fixed bit depth parsing for files from newer versions of HCImage
- **SimplePCI/HCImage .cxd**
    * fixed image dimensions to allow for extra padding bytes
- **Leica LIF**
    * improved reading of image descriptions
- **ICS**
    * fixed to use correct units for timestamps and physical pixel sizes
- **MicroManager**
    * fixed to use correct units for timestamps
- **Gatan .dm3/.dm4**
    * fixed problems with reading double-precision metadata values
- **Hamamatsu NDPI**
    * fixed reading of mask images
- **Leica .lei**
    * fixed reading of bit depth and endianness for datasets that were modified after acquisition
- **FEI TIFF**
    * updated to read metadata from files produced by FEI Titan systems
- **QuickTime**
    * fixed to handle planes with no stored pixels
- **Leica .scn**
    * fixed reading of files that contain fewer images than expected
- **Zeiss .czi**
    * fixed channel colors when an alpha value is not recorded
    * fixed handling of pre-stitched image tiles
- **SDT**
    * added support for Zip-compressed images
- **Nikon .nd2**
    * fixed to read image dimensions from new non-XML metadata
- **OME-XML**
    * fixed writing of integer metadata values

- **Native C++ updates:**
  - completed support for building on Windows
- **Documentation updates, including:**
  - updated instructions for running automated data tests
  - clarified JVM versions currently supported

5.74 5.1.3 (2015 July 21)

- **Native C++ updates:**
  - Added cmake superbuild to build core dependencies (zlib, bzip2, png, icu, xerces, boost)
  - Progress on support for Windows
- **Bug fixes, including:**
  - Fixed segfault in the *showinf* tool used with the C++ bindings
  - Allow reading from https URLs
  - **ImageJ**
    - improved performance of displaying ROIs
  - **Command line tools**
    - fixed bfconvert to correctly create datasets with multiple files
  - **Metamorph**
    - improved detection of time series
    - fixed .nd datasets with variable Z and T counts in each channel
    - fixed .nd datasets that contain invalid TIFF/STK files
    - fixed dimensions when the number of planes does not match the recorded Z, C, and T sizes
  - **SlideBook**
    - improved native library detection (thanks to Richard Myers)
  - **JPEG**
    - fixed decompression of lossless files with multiple channels (thanks to Aaron Avery)
  - **Imspector OBF**
    - updated to support version 2 files (thanks to Bjoern Thiel)
  - **Imspector MSR**
    - improved detection of Z stacks
  - **PerkinElmer Opera Flex**
    - improved handling of multiple acquisitions of the same plate
  - **Zeiss CZI**
    - fixed error when opening single-file datasets whose names contained “(” and “)”
  - **TIFF**
    - improved speed of reading files with many tiles

  - **AVI**
    - updated to read frame index (idx1) tables
  - **Nikon ND2**
    - fixed channel counts for files with more than 3 channels
  - **PNG**
    - fixed decoding of interlaced images with a width or height that is not a multiple of 8
  - **PSD**
    - improved reading of compressed images
- **Documentation improvements, including:**
  - updated instructions for writing a new file format reader
  - updated usage information for command line tools
  - new Javadocs for the *MetadataStore* and *MetadataRetrieve* interfaces

5.75 5.1.2 (2015 May 28)

- Added OME-TIFF writing support to the native C++ implementation
- OME-TIFF export: switch to BigTIFF if .ome.tf2, .ome.tf8, or .ome.btf extensions are used
- Improved MATLAB developer documentation
- Added SlideBook reader that uses the SDK from 3I (thanks to Richard Myers and 3I - Intelligent Imaging Innovations)
- Preliminary work to make MATLAB toolbox work with Octave
- **Many bug fixes, including:**
  - **ImageJ**
    - fixed regression in getPlanePosition* macro extension methods
    - fixed display of composite color virtual stacks
  - **Nikon ND2**
    - improved parsing of plane position and timestamp data
  - **TIFF**
    - reduced memory required to read color lookup tables
  - **Zeiss LSM**
    - improved parsing of 16-bit color lookup tables
  - **Zeiss CZI**
    - fixed ordering of original metadata table
    - fixed reading of large pre-stitched tiled images
  - **AIM**
    - fixed handling of truncated files
  - **Metamorph/MetaXpress TIFF**

        * improved UIC1 metadata tag parsing

5.76 5.1.1 (2015 April 28)

- Add TIFF writing support to the native C++ implementation
- Fixed remaining functional differences between Windows and Mac/Linux
- Improved performance of ImageJ plugin when working with ROIs
- TIFF export: switch to BigTIFF if .tf2, .tf8, or .btf extensions are used
- **Many bug fixes, including:**
    - fixed upgrade checking to more accurately report when a new version is available
    - **Zeiss CZI**
        * fixed ordering of multiposition data
        * improved support for RGB and fused images
    - **Nikon ND2**
        * improved ordering of multiposition data
    - **Leica LIF**
        * improved metadata validity checks
        * improved excitation wavelength detection
    - **Metamorph STK/TIFF**
        * record lens numerical aperture
        * fixed millisecond values in timestamps
    - **Gatan DM3**
        * correctly detect signed pixel data
    - **Imaris HDF**
        * fix channel count detection
    - **ICS export**
        * fix writing of files larger than 2GB

5.77 5.1.0 (2015 April 2)

- Improvements to performance with network file systems
- Improvements to developer documentation
- Initial version of native C++ implementation
- Improved support for opening and saving ROI data with ImageJ
- Added support for *CellH5* data (thanks to Christoph Sommer)
- Added support for *Perkin Elmer Nuance* data (thanks to Lee Kamentsky)
- Added support for *Amnis FlowSight* data (thanks to Lee Kamentsky and Sebastien Simard)
- Added support for *Veeco AFM* data

- Added support for *Zeiss .lms* data (not to be confused with .lsm)
- Added support for *I2I* data
- Added support for writing Vaa3D data (thanks to Brian Long)
- Updated to OME schema 2015-01
- Update RandomAccessInputStream and RandomAccessOutputStream to read and write bits
- **Many bug fixes, including:**
  - **Leica SCN**
    - fix pixel data decompression
    - fix handling of files with multiple channels
    - parse magnification and physical pixel size data
  - **Olympus/CellSens .vsi**
    - more thorough parsing of metadata
    - improved reading of thumbnails and multi-resolution images
  - **NDPI**
    - fix reading of files larger than 4GB
    - parse magnification data
  - **Zeiss CZI**
    - improve parsing of plane position coordinates
  - **Inveon**
    - fix reading of files larger than 2 GB
  - **Nikon ND2**
    - many improvements to dimension detection
    - many improvements to metadata parsing accuracy
    - update original metadata table to include PFS data
  - **Gatan DM3**
    - fix encoding when parsing metadata
    - fix physical pixel size parsing
  - **Metamorph**
    - fix off-by-one in metadata parsing
    - fix number parsing to be independent of the system locale
  - **JPEG**
    - parse EXIF data, if present (thanks to Paul Van Schayck)
  - **OME-XML/OME-TIFF**
    - fix handling of missing image data
  - **PrairieView**
    - improved support for version 5.2 data (thanks to Curtis Rueden)

  - **DICOM**
    - fix dimensions for multi-file datasets
    - fix pixel data decoding for files with multiple images
  - **PNG**
    - reduce memory required to read large images
  - **Imspector OBF**
    - fix support for version 5 data (thanks to Bjoern Thiel)
  - **PCORAW**
    - fix reading of files larger than 4 GB
  - **AIM**
    - fix reading of files larger than 4 GB
  - **MRC**
    - add support for signed 8-bit data
  - Fix build errors in MIPAV plugin
  - **ImageJ**
    - fix export from a script/macro
    - fix windowless export
    - allow exporting from any open image window
    - allow the "Group files with similar names" and "Swap dimensions" options to be used from a script/macro
  - **bfconvert**
    - fix writing each channel, Z section, and/or timepoint to a separate file
    - add options for configuring the tile size to be used when saving images

5.78 5.0.8 (2015 February 10)

- No changes - release to keep version numbers in sync with OMERO

5.79 5.0.7 (2015 February 5)

- **Several bug fixes, including:**
  - ND filter parsing for DeltaVision
  - Timepoint count and original metadata parsing for Metamorph
  - Build issues when Genshi or Git are missing
  - LZW image decoding

5.80 5.0.6 (2014 November 11)

- **Several bug fixes, including:**
  - Pixel sign for DICOM images
  - Image dimensions for Zeiss CZI and Nikon ND2
  - Support for Leica LIF files produced by LAS AF 4.0 and later

5.81 5.0.5 (2014 September 23)

- Documentation improvements
- Support for non-spectral Prairie 5.2 datasets

5.82 5.0.4 (2014 September 3)

- Fix compile and runtime errors under Java 1.8
- Improvements to Nikon .nd2 metadata parsing
- Added support for PicoQuant .bin files (thanks to Ian Munro)

5.83 5.0.3 (2014 August 7)

- Many bug fixes for Nikon .nd2 files
- **Several other bug fixes, including:**
  - LZW image decoding
  - Stage position parsing for Zeiss CZI
  - Exposure time units for ScanR
  - Physical pixel size units for DICOM
  - NDPI and Zeiss LSM files larger than 4GB
  - Z and T dimensions for InCell 6000 plates
  - Export of RGB images in ImageJ
- Improved metadata saving in MATLAB functions

5.84 5.0.2 (2014 May 28)

- Many bug fixes for Zeiss .czi files
- **Several other bug fixes, including:**
  - Gatan .dm3 units and step count parsing
  - Imspector .msr 5D image support
  - DICOM reading of nested tags
- Update native-lib-loader version (to 2.0.1)
- Updates and improvements to user documentation

5.85 5.0.1 (2014 Apr 7)

- Added image pyramid support for CellSens .vsi data
- **Several bug fixes, including:**
  - Woolz import into OMERO
  - Cellomics file name parsing (thanks to Lee Kamentsky)
  - Olympus FV1000 timestamp support (thanks to Lewis Kraft and Patrick Riley)
  - (A)PNG large image support
  - Zeiss .czi dimension detection for SPIM datasets
- Performance improvements for Becker & Hickl .sdt file reading (thanks to Ian Munro)
- Performance improvements to directory listing over NFS
- Update slf4j and logback versions (to 1.7.6 and 1.1.1 respectively)
- Update jgoodies-forms version (to 1.7.2)

5.86 5.0.0 (2014 Feb 25)

- New bundled 'bioformats_package.jar' for ImageJ
- Now uses logback as the slf4j binding by default
- Updated component names, .jar file names, and Maven artifact names
- Fixed support for Becker & Hickl .sdt files with multiple blocks
- Fixed tiling support for TIFF, Hamamatsu .ndpi, JPEG, and Zeiss .czi files
- Improved continuous integration testing
- Updated *command line documentation*

5.87 5.0.0-RC1 (2013 Dec 19)

- Updated Maven build system and launched new Artifactory repository (https://artifacts.openmicroscopy.org)
- **Added support for:**
  - *Bio-Rad SCN*
  - *Yokogawa CellVoyager* (thanks to Jean-Yves Tinevez)
  - *LaVision Imspector*
  - *PCORAW*
  - Woolz (thanks to Bill Hill)
- Added support for populating and parsing ModuloAlong{Z, C, T} annotations for FLIM/SPIM data
- Updated netCDF and slf4j version requirements - netCDF 4.3.19 and slf4j 1.7.2 are now required
- Updated and improved *MATLAB users* and *developers* documentation
- Many bug fixes including for Nikon ND2, Zeiss CZI, and CellWorX formats

5.88 5.0.0-beta1 (2013 June 20)

- Updated to 2013-06 OME-XML schema
- Improved the performance in tiled formats
- Added caching of Reader metadata using https://github.com/EsotericSoftware/kryo
- **Added support for:**
  - *Aperio AFI*
  - *Inveon*
  - *MPI-BPC Imspector*
- **Many bug fixes, including:**
  - Add ZEN 2012/Lightsheet support to Zeiss CZI
  - Improved testing of autogenerated code
  - Moved OME-XML specification into Bio-Formats repository

5.89 4.4.10 (2014 Jan 15)

- Bug fixes including CellWorx, Metamorph and Zeiss CZI
- Updates to MATLAB documentation

5.90 4.4.9 (2013 Oct 16)

- Many bug fixes including improvements to support for ND2 format
- Java 1.6 is now the minimum supported version; Java 1.5 is no longer supported

5.91 4.4.8 (2013 May 2)

- No changes - release to keep version numbers in sync with OMERO

5.92 4.4.7 (2013 April 25)

- Many bug fixes to improve support for more than 20 formats
- Improved export to multi-file datasets
- Now uses slf4j for logging rather than using log4j directly, enabling other logging implementations to be used, for example when Bio-Formats is used as a component in other software using a different logging system.

5.93 4.4.6 (2013 February 11)

- Many bug fixes
- Further documentation improvements

5.94 4.4.5 (2012 November 13)

- Restructured and improved documentation
- **Many bug fixes, including:**
  - File grouping in many multi-file formats
  - Maven build fixes
  - ITK plugin fixes

5.95 4.4.4 (2012 September 24)

- Many bug fixes

5.96 4.4.2 (2012 August 22)

- Security fix for OMERO plugins for ImageJ

5.97 4.4.1 (2012 July 20)

- Fix a bug that prevented BigTIFF files from being read
- Fix a bug that prevented PerkinElmer .flex files from importing into OMERO

5.98 4.4.0 (2012 July 13)

- Many, many bug fixes
- **Added support for:**
  - .nd2 files from Nikon Elements version 4
  - PerkinElmer Operetta data
  - MJPEG-compressed AVIs
  - MicroManager datasets with multiple positions
  - Zeiss CZI data
  - IMOD data

5.99 4.3.3 (2011 October 18)

- **Many bug fixes, including:**
  - Speed improvements to HCImage/SimplePCI and Zeiss ZVI files
  - Reduce memory required by Leica LIF reader
  - More accurately populate metadata for Prairie TIFF datasets
  - Various fixes to improve the security of the OMERO plugin for ImageJ
  - Better dimension detection for Bruker MRI datasets
  - Better thumbnail generation for histology (SVS, NDPI) datasets

  - Fix stage position parsing for Metamorph TIFF datasets
  - Correctly populate the channel name for PerkinElmer Flex files

5.100 4.3.2 (2011 September 15)

- **Many bug fixes, including:**
  - Better support for Volocity datasets that contain compressed data
  - More accurate parsing of ICS metadata
  - More accurate parsing of cellSens .vsi files
- **Added support for a few new formats**
  - .inr
  - Canon DNG
  - Hitachi S-4800
  - Kodak .bip
  - JPX
  - Volocity Library Clipping (.acff)
  - Bruker MRI
- Updated Zeiss LSM reader to parse application tags
- Various performance improvements, particularly for reading/writing TIFFs
- Updated OMERO ImageJ plugin to work with OMERO 4.3.x

5.101 4.3.1 (2011 July 8)

- **Several bug fixes, including:**
  - Fixes for multi-position DeltaVision files
  - Fixes for MicroManager 1.4 data
  - Fixes for 12 and 14-bit JPEG-2000 data
  - Various fixes for reading Volocity .mvd2 datasets
- Added various options to the 'showinf' and 'bfconvert' command line tools
- Added better tests for OME-XML backwards compatibility
- Added the ability to roughly stitch tiles in a multi-position dataset

5.102 4.3.0 (2011 June 14)

- **Many bug fixes, including:**
  - Many fixes for reading and writing sub-images
  - Fixes for stage position parsing in the Zeiss formats
  - File type detection fixes

- Updated JPEG-2000 reading and writing support to be more flexible
- **Added support for 9 new formats:**
  - InCell 3000
  - Trestle
  - Hamamatsu .ndpi
  - Hamamatsu VMS
  - SPIDER
  - Volocity .mvd2
  - Olympus SIS TIFF
  - IMAGIC
  - cellSens VSI
- Updated to 2011-06 OME-XML schema
- Minor speed improvements in many formats
- Switched version control system from SVN to Git
- Moved all Trac tickets into the OME Trac: https://trac.openmicroscopy.org
- Improvements to testing frameworks
- Added Maven build system as an alternative to the existing Ant build system
- Added pre-compiled C++ bindings to the download page

5.103 4.2.2 (2010 December 6)

- **Several bug fixes, notably:**
  - Metadata parsing fixes for Zeiss LSM, Metamorph STK, and FV1000
  - Prevented leaked file handles when exporting to TIFF/OME-TIFF
  - Fixed how BufferedImages are converted to byte arrays
- Proper support for OME-XML XML annotations
- Added support for SCANCO Medical .aim files
- Minor improvements to ImageJ plugins
- Added support for reading JPEG-compressed AVI files

5.104 4.2.1 (2010 November 12)

- Many, many bug fixes
- **Added support for 7 new formats:**
  - CellWorX .pnl
  - ECAT7
  - Varian FDF
  - Perkin Elmer Densitometer

  - FEI TIFF
  - Compix/SimplePCI TIFF
  - Nikon Elements TIFF
- Updated Zeiss LSM metadata parsing, with generous assistance from Zeiss, FMI, and MPI-CBG
- Lots of work to ensure that converted OME-XML validates
- Improved file stitching functionality; non-numerical file patterns and limited regular expression-style patterns are now supported

5.105 4.2.0 (2010 July 9)

- Fixed many, many bugs in all aspects of Bio-Formats
- Reworked ImageJ plugins to be more user- and developer-friendly
- Added many new unit tests
- Added support for approximately 25 new file formats, primarily in the SPM domain
- Rewrote underlying I/O infrastructure to be thread-safe and based on Java NIO
- Rewrote OME-XML parsing/generation layer; OME-XML 2010-06 is now supported
- Improved support for exporting large images
- Improved support for exporting to multiple files
- Updated logging infrastructure to use slf4j and log4j

5.106 4.1.1 (2009 December 3)

- Fixed many bugs in popular file format readers

4.1 (2009 October 21):

- Fixed many bugs in most file format readers
- Significantly improved confocal and HCS metadata parsing
- Improved C++ bindings
- Eliminated references to Java AWT classes in core Bio-Formats packages
- Added support for reading Flex datasets from multiple servers
- Improved OME-XML generation; generated OME-XML is now valid
- Added support for Olympus ScanR data
- Added OSGi information to JARs
- Added support for Amira Mesh files
- Added support for LI-FLIM files
- Added more informative exceptions
- Added support for various types of ICS lifetime data
- Added support for Nikon EZ-C1 TIFFs
- Added support for Maia Scientific MIAS data

5.107 4.0.1 (2009 June 1)

- Lots of bug fixes in most format readers and writers
- Added support for Analyze 7.1 files
- Added support for Nifti files
- Added support for Cellomics .c01 files
- Refactored ImageJ plugins
- Bio-Formats, the common package, and the ImageJ plugins now require Java 1.5
- Eliminated native library dependency for reading lossless JPEGs
- Changed license from GPL v3 or later to GPL v2 or later
- Updated Olympus FV1000, Zeiss LSM, Zeiss ZVI and Nikon ND2 readers to parse ROI data
- Added option to ImageJ plugin for displaying ROIs parsed from the chosen dataset
- Fixed BufferedImage construction for signed data and unsigned int data

5.108 4.0.0 (2009 March 3)

- Improved OME data model population for Olympus FV1000, Nikon ND2, Metamorph STK, Leica LEI, Leica LIF, InCell 1000 and MicroManager
- Added TestNG tests for format writers
- Added option to ImageJ plugin to specify custom colors when customizing channels
- Added ability to upgrade the ImageJ plugin from within ImageJ
- Fixed bugs in Nikon ND2, Leica LIF, BioRad PIC, TIFF, PSD, and OME-TIFF
- Fixed bugs in Data Browser and Exporter plugins
- Added support for Axon Raw Format (ARF), courtesy of Johannes Schindelin
- Added preliminary support for IPLab-Mac file format

5.109 2008 December 29

- Improved metadata support for DeltaVision, Zeiss LSM, MicroManager, and Leica LEI
- Restructured code base/build system to be component-driven
- Added support for JPEG and JPEG-2000 codecs within TIFF, OME-TIFF and OME-XML
- Added support for 16-bit compressed Flex files
- Added support for writing JPEG-2000 files
- Added support for Minolta MRW format
- Added support for the 2008-09 release of OME-XML
- Removed dependency on JMagick
- Re-added caching support to data browser plugin
- Updated loci.formats.Codec API to be more user-friendly
- Expanded loci.formats.MetadataStore API to better represent the OME-XML model

- Improved support for Nikon NEF
- Improved support for TillVision files
- Improved ImageJ import options dialog
- Fixed bugs with Zeiss LSM files larger than 4 GB
- Fixed minor bugs in most readers
- Fixed bugs with exporting from an Image5D window
- Fixed several problems with virtual stacks in ImageJ

5.110 2008 August 30

- Fixed bugs in many file format readers
- Fixed several bugs with swapping dimensions
- Added support for Olympus CellR/APL files
- Added support for MINC MRI files
- Added support for Aperio SVS files compressed with JPEG 2000
- Added support for writing OME-XML files
- Added support for writing APNG files
- Added faster LZW codec
- Added drag and drop support to ImageJ shortcut window
- Re-integrated caching into the data browser plugin

5.111 2008 July 1

- Fixed bugs in most file format readers
- Fixed bugs in OME and OMERO download functionality
- Fixed bugs in OME server-side import
- Improved metadata storage/retrieval when uploading to and downloading from the OME Perl server
- Improved Bio-Formats ImageJ macro extensions
- Major updates to MetadataStore API
- Updated OME-XML generation to use 2008-02 schema by default
- Addressed time and memory performance issues in many readers
- Changed license from LGPL to GPL
- Added support for the FEI file format
- Added support for uncompressed Hamamatsu Aquacosmos NAF files
- Added support for Animated PNG files
- Added several new options to Bio-Formats ImageJ plugin
- Added support for writing ICS files

5.112 2008 April 17

- Fixed bugs in Slidebook, ND2, FV1000 OIB/OIF, Perkin Elmer, TIFF, Prairie, Openlab, Zeiss LSM, MNG, Molecular Dynamics GEL, and OME-TIFF
- Fixed bugs in OME and OMERO download functionality
- Fixed bugs in OME server-side import
- Fixed bugs in Data Browser
- Added support for downloading from OMERO 2.3 servers
- Added configuration plugin
- Updates to MetadataStore API
- Updates to OME-XML generation - 2007-06 schema used by default
- Added support for Li-Cor L2D format
- Major updates to TestNG testing framework
- Added support for writing multi-series OME-TIFF files
- Added support for writing BigTIFF files

5.113 2008 Feb 12

- Fixed bugs in QuickTime, SimplePCI and DICOM
- Fixed a bug in channel splitting logic

5.114 2008 Feb 8

- Many critical bugfixes in format readers and ImageJ plugins
- **Newly reborn Data Browser for 5D image visualization**
  - some combinations of import options do not work yet

5.115 2008 Feb 1

- Fixed bugs in Zeiss LSM, Metamorph STK, FV1000 OIB/OIF, Leica LEI, TIFF, Zeiss ZVI, ICS, Prairie, Openlab LIFF, Gatan, DICOM, QuickTime
- Fixed bug in OME-TIFF writer
- Major changes to MetadataStore API
- Added support for JPEG-compressed TIFF files
- **Added basic support for Aperio SVS files**
  - JPEG2000 compression is still not supported
- Improved "crop on import" functionality
- Improvements to bfconvert and bfview
- Improved OME-XML population for several formats
- Added support for JPEG2000-compressed DICOM files

- EXIF data is now parsed from TIFF files

5.116 2007 Dec 28

- Fixed bugs in Leica LEI, Leica TCS, SDT, Leica LIF, Visitech, DICOM, Imaris 5.5 (HDF), and Slidebook readers
- Better parsing of comments in TIFF files exported from ImageJ
- Fixed problem with exporting 48-bit RGB data
- Added logic to read multi-series datasets spread across multiple files
- Improved channel merging in ImageJ - requires ImageJ 1.39l
- Support for hyperstacks and virtual stacks in ImageJ - requires ImageJ 1.39l
- Added API for reading directly from a byte array or InputStream
- Metadata key/value pairs are now stored in ImageJ's "Info" property
- Improved OMERO download plugin - it is now much faster
- Added "open all series" option to ImageJ importer
- ND2 reader based on Nikon's SDK now uses our own native bindings
- Fixed metadata saving bug in ImageJ
- Added sub-channel labels to ImageJ windows
- Major updates to 4D Data Browser
- Minor updates to automated testing suite

5.117 2007 Dec 1

- Updated OME plugin for ImageJ to support downloading from OMERO
- Fixed bug with floating point TIFFs
- Fixed bugs in Visitech, Zeiss LSM, Imaris 5.5 (HDF)
- Added alternate ND2 reader that uses Nikon's native libraries
- Fixed calibration and series name settings in importer
- Added basic support for InCell 1000 datasets

5.118 2007 Nov 21

- Fixed bugs in ND2, Leica LIF, DICOM, Zeiss ZVI, Zeiss LSM, FV1000 OIB, FV1000 OIF, BMP, Evotec Flex, BioRad PIC, Slidebook, TIFF
- Added new ImageJ plugins to slice stacks and do "smart" RGB merging
- **Added "windowless" importer plugin**
  - uses import parameters from IJ_Prefs.txt, without prompting the user
- Improved stack slicing and colorizing logic in importer plugin
- **Added support for DICOM files compressed with lossless JPEG**
  - requires native libraries

- Fixed bugs with signed pixel data
- Added support for Imaris 5.5 (HDF) files
- Added 4 channel merging to importer plugin
- Added API methods for reading subimages
- Major updates to the 4D Data Browser

5.119 2007 Oct 17

- Critical OME-TIFF bugfixes
- Fixed bugs in Leica LIF, Zeiss ZVI, TIFF, DICOM, and AVI readers
- Added support for JPEG-compressed ZVI images
- Added support for BigTIFF
- Added importer plugin option to open each plane in a new window
- Added MS Video 1 codec for AVI

5.120 2007 Oct 1

- Added support for compressed DICOM images
- Added support for uncompressed LIM files
- Added support for Adobe Photoshop PSD files
- Fixed bugs in DICOM, OME-TIFF, Leica LIF, Zeiss ZVI, Visitech, PerkinElmer and Metamorph
- Improved indexed color support
- Addressed several efficiency issues
- Fixed how multiple series are handled in 4D data browser
- Added option to reorder stacks in importer plugin
- Added option to turn off autoscaling in importer plugin
- Additional metadata convenience methods

5.121 2007 Sept 11

- Major improvements to ND2 support; lossless compression now supported
- Support for indexed color images
- Added support for Simple-PCI .cxd files
- Command-line OME-XML validation
- Bugfixes in most readers, especially Zeiss ZVI, Metamorph, PerkinElmer and Leica LEI
- Initial version of Bio-Formats macro extensions for ImageJ

5.122 2007 Aug 1

- Added support for latest version of Leica LIF
- Fixed several issues with Leica LIF, Zeiss ZVI
- Better metadata mapping for Zeiss ZVI
- Added OME-TIFF writer
- Added MetadataRetrieve API for retrieving data from a MetadataStore
- Miscellaneous bugfixes

5.123 2007 July 16

- Fixed several issues with ImageJ plugins
- Better support for Improvision and Leica TCS TIFF files
- Minor improvements to Leica LIF, ICS, QuickTime and Zeiss ZVI readers
- Added searchable metadata window to ImageJ importer

5.124 2007 July 2

- Fixed issues with ND2, Openlab LIFF and Slidebook
- Added support for Visitech XYS
- Added composite stack support to ImageJ importer

5.125 2007 June 18

- Fixed issues with ICS, ND2, MicroManager, Leica LEI, and FV1000 OIF
- Added support for large (> 2 GB) ND2 files
- Added support for new version of ND2
- Minor enhancements to ImageJ importer
- Implemented more flexible logging
- Updated automated testing framework to use TestNG
- Added package for caching images produced by Bio-Formats

5.126 2007 June 6

- Fixed OME upload/download bugs
- Fixed issues with ND2, EPS, Leica LIF, and OIF
- Added support for Khoros XV
- Minor improvements to the importer

5.127 2007 May 24

- Better Slidebook support
- Added support for Quicktime RPZA
- Better Leica LIF metadata parsing
- Added support for BioRad PIC companion files
- Added support for bzip2-compressed files
- Improved ImageJ plugins
- Native support for FITS and PGM

5.128 2007 May 2

- Added support for NRRD
- Added support for Evotec Flex (requires LuraWave Java SDK with license code)
- Added support for gzip-compressed files
- Added support for compressed QuickTime headers
- Fixed QuickTime Motion JPEG-B support
- Fixed some memory issues (repeated small array allocations)
- Fixed issues reading large (> 2 GB) files
- Removed "ignore color table" logic, and replaced with Leica-specific solution
- Added status event reporting to readers
- Added API to toggle metadata collection
- Support for multiple dimensions rasterized into channels
- Deprecated reader and writer methods that accept the 'id' parameter
- Deprecated IFormatWriter.save in favor of saveImage and saveBytes
- Moved dimension swapping and min/max calculation logic to delegates
- Separate GUI logic into isolated loci.formats.gui package
- Miscellaneous bugfixes and tweaks in most readers and writers
- Many other bugfixes and improvements

5.129 2007 Mar 16

- Fixed calibration bugs in importer plugin
- Enhanced metadata support for additional formats
- Fixed LSM bug

5.130 2007 Mar 7

- Added support for Micro-Manager file format
- Fixed several bugs – Leica LIF, Leica LEI, ICS, ND2, and others
- Enhanced metadata support for several formats
- Load series preview thumbnails in the background
- Better implementation of openBytes(String, int, byte[]) for most readers
- Expanded unit testing framework

5.131 2007 Feb 28

- Better series preview thumbnails
- Fixed bugs with multi-channel Leica LEI
- Fixed bugs with "ignore color tables" option in ImageJ plugin

5.132 2007 Feb 26

- Many bugfixes: Leica LEI, ICS, FV1000 OIB, OME-XML and others
- Better metadata parsing for BioRad PIC files
- Enhanced API for calculating channel minimum and maximum values
- Expanded MetadataStore API to include more semantic types
- Added thumbnails to series chooser in ImageJ plugin
- Fixed plugins that upload and download from an OME server

5.133 2007 Feb 7

- Added plugin for downloading images from OME server
- Improved HTTP import functionality
- Added metadata filtering – unreadable metadata is no longer shown
- Better metadata table for multi-series datasets
- Added support for calibration information in Gatan DM3
- Eliminated need to install JAI Image I/O Tools to read ND2 files
- Fixed ZVI bugs: metadata truncation, and other problems
- Fixed bugs in Leica LIF: incorrect calibration, first series labeling
- Fixed memory bug in Zeiss LSM
- Many bugfixes: PerkinElmer, DeltaVision, Leica LEI, LSM, ND2, and others
- IFormatReader.close(boolean) method to close files temporarily
- Replaced Compression utility class with extensible Compressor interface
- Improved testing framework to use .bioformats configuration files

5.134 2007 Jan 5

- Added support for Prairie TIFF
- Fixed bugs in Zeiss LSM, OIB, OIF, and ND2
- Improved API for writing files
- Added feature to read files over HTTP
- Fixed bugs in automated testing framework
- Miscellaneous bugfixes

5.135 2006 Dec 22

- Expanded ImageJ plugin to optionally use Image5D or View5D
- Improved support for ND2 and JPEG-2000 files
- Added automated testing framework
- Fixed bugs in Zeiss ZVI reader
- Miscellaneous bugfixes

5.136 2006 Nov 30

- Added support for ND2/JPEG-2000
- Added support for MRC
- Added support for MNG
- Improved support for floating-point images
- Fixed problem with 2-channel Leica LIF data
- Minor tweaks and bugfixes in many readers
- Improved file stitching logic
- Allow ImageJ plugin to be called from a macro

5.137 2006 Nov 2

- Bugfixes and improvements for Leica LIF, Zeiss LSM, OIF and OIB
- Colorize channels when they are split into separate windows
- Fixed a bug with 4-channel datasets

5.138 2006 Oct 31

- Added support for Imaris 5 files
- Added support for RGB ICS images

5.139 2006 Oct 30

- Added support for tiled TIFFs
- Fixed bugs in ICS reader
- Fixed importer plugin deadlock on some systems

5.140 2006 Oct 27

- Multi-series support for Slidebook
- Added support for Alicona AL3D
- Fixed plane ordering issue with FV1000 OIB
- Enhanced dimension detection in FV1000 OIF
- Added preliminary support for reading NEF images
- Added option to ignore color tables
- Fixed ImageJ GUI problems
- Fixed spatial calibration problem in ImageJ
- Fixed some lingering bugs in Zeiss ZVI support
- Fixed bugs in OME-XML reader
- Tweaked ICS floating-point logic
- Fixed memory leaks in all readers
- Better file stitching logic

5.141 2006 Oct 6

- Support for 3i SlideBook format (single series only for now)
- Support for 16-bit RGB palette TIFF
- Fixed bug preventing import of certain Metamorph STK files
- Fixed some bugs in PerkinElmer UltraView support
- Fixed some bugs in Leica LEI support
- Fixed a bug in Zeiss ZVI support
- Fixed bugs in Zeiss LSM support
- Fixed a bug causing slow identification of Leica datasets
- Fixed bugs in the channel merging logic
- Fixed memory leak for OIB format
- Better scaling of 48-bit RGB data to 24-bit RGB
- Fixed duplicate channels bug in “open each channel in a separate window”
- Fixed a bug preventing PICT import into ImageJ
- Better integration with HandleExtraFileTypes

- Better virtual stack support in Data Browser plugin
- Fixed bug in native QuickTime random access
- Keep aspect ratio for computed thumbnails
- Much faster file stitching logic

5.142 2006 Sep 27

- PerkinElmer: support for PE UltraView
- Openlab LIFF: support for Openlab v5
- Leica LEI: bugfixes, and support for multiple series
- ZVI, OIB, IPW: more robust handling of these formats (eliminated custom OLE parsing logic in favor of Apache POI)
- OIB: better metadata parsing (but maybe still not perfect?)
- LSM: fixed a bug preventing import of certain LSMs
- Metamorph STK: fixed a bug resulting in duplicate image planes
- User interface: use of system look & feel for file chooser dialog when available
- Better notification when JAR libraries are missing

5.143 2006 Sep 6

- Leica LIF: multiple distinct image series within a single file
- Zeiss ZVI: fixes and improvements contributed by Michel Boudinot
- Zeiss LSM: fixed bugs preventing the import of certain LSM files
- TIFF: fixed a bug preventing import of TIFFs created with Bio-Rad software

5.144 2006 Mar 31

- First release

CHAPTER

SIX

REPORTING A BUG

6.1 Before filing a bug report

If you think you have found a bug in Bio-Formats, the first thing to do is update your version of Bio-Formats to the latest version to check if the problem has already been addressed. The Fiji updater will automatically do this for you, while in ImageJ you can select *Plugins* → *Bio-Formats* → *Update Bio-Formats Plugins*.

You can also download the latest version of Bio-Formats from the OME website.

6.2 Common issues to check

- If you get an error message similar to:

```
java.lang.UnsupportedClassVersionError: loci/plugins/LociImporter :
Unsupported major.minor version 52.0

This plugin requires Java 1.8 or later.
```

  you need to upgrade your system Java version to Java 8 or above, or download a new version of ImageJ/Fiji bundled with Java 8.

- If your 12, 14 or 16-bit images look all black when you open them, typically the problem is that the pixel values are very, very small relative to the maximum possible pixel value (4095, 16383, and 65535, respectively), so when displayed the pixels are effectively black. In ImageJ/Fiji, this is fixable by checking the "Autoscale" option; with the command line tools, the "-autoscale -fast" options should work.

- If the file is very, very small (4096 bytes) and any exception is generated when reading the file, then make sure it is not a Mac OS X resource fork. The 'file' command should tell you:

```
$ file /path/to/suspicious-file
suspicious-file: AppleDouble encoded Macintosh file
```

- If you get an `OutOfMemory` or `NegativeArraySize` error message when attempting to open an SVS or JPEG-2000 file then the amount of pixel data in a single image plane exceeds the amount of memory allocated to the JVM or 2 GB, respectively. For the former, you can increase the amount of memory allocated; in the latter case, you will need to open the image in sections. If you are using Bio-Formats as a library, this means using the `openBytes(int, int, int, int, int)` method in loci.formats.IFormatReader. If you are using Bio-Formats within ImageJ, you can use the *Crop on import* option.

  Note that JPEG-2000 is a very efficient compression algorithm - thus the size of the file on disk will be substantially smaller than the amount of memory required to store the uncompressed pixel data. It is not uncommon for a JPEG-2000 or SVS file to occupy less than 200 MB on disk, and yet have over 2 GB of uncompressed pixel data.

6.3 Sending a bug report

If you can still reproduce the bug after updating to the latest version of Bio-Formats, and your issue does not relate to anything listed above or noted on the relevant file format page, please send a bug report to the forums. You can upload sample files to the Bio-Formats Zenodo community, or for files over 50 GB, we can provide you with an FTP server address.

To ensure that any inquiries you make are resolved promptly, please include the following information:

- **Exact error message**. Copy and paste any error messages into the text of your email. Alternatively, attach a screenshot of the relevant windows.
- **Version information**. Indicate which release of Bio-Formats, which operating system, and which version of Java you are using.
- **Non-working data**. If possible, please send a non-working file that may be made public under a liberal license, at least CC-BY 4.0. This helps us ensure that the problem is fixed for next release and will not reappear in later releases. Providing data that cannot be made public is strongly discouraged. See Receipt of Sample Data.
- **Metadata and screenshots**. If possible, include any additional information about your data. We are especially interested in the expected dimensions (width, height, number of channels, Z slices, and timepoints). Screenshots of the image being successfully opened in other software are also useful.
- **Format details**. If you are requesting support for a new format, we ask that you send as much data as you have regarding this format (sample files, specifications, vendor/manufacturer information, etc.). This helps us to better support the format and ensures future versions of the format are also supported.

Please be patient - it may be a few days until you receive a response, but we reply to *every* email inquiry we receive.

INDEX

Symbols

.1sc, 118
.2, 183
.2fl, 198
.3, 183
.4, 183
.acff, 204
.afi, 112
.afm, 189
.aim, 108
.al3d, 108
.ali, 168
.am, 110
.amiramesh, 110
.apl, 173
.aqproj, 198
.arf, 116
.avi, 115
.bif, 202
.bin, 186
.bip, 154
.bmp, 138, 206
.btf, 196
.c01, 124
.cfg, 188
.ch5, 124
.cif, 110
.cr2, 123
.crw, 123
.csv, 182
.cxd, 192
.czi, 209
.dat, 150, 176, 200
.db, 195
.dcimg, 135
.dcm, 127
.dib, 124
.dicom, 127
.dm2, 133
.dm3, 132
.dm4, 132
.dti, 203
.dv, 127
.eps, 129
.epsi, 129
.exp, 116
.fdf, 200
.fff, 140
.ffr, 198
.fits, 132
.flex, 130
.fli, 155
.frm, 146
.gel, 109
.gif, 134
.grey, 110
.h5, 118
.hdf, 202
.hdr, 111, 147, 170, 200
.hed, 141
.his, 136
.htd, 114, 163
.html, 203
.hx, 110
.i2i, 138
.ics, 139
.ids, 139
.im3, 182
.img, 111, 122, 130, 141, 150, 170
.ims, 121
.inr, 146
.ipl, 148
.ipm, 149
.ipw, 141
.j2k, 152
.jdce, 150
.jp2, 152
.jpf, 152
.jpg, 138, 151, 199
.jpk, 152
.jpx, 153
.klb, 153
.l2d, 161
.labels, 110

.lei, 156
.lif, 157
.liff, 143
.lim, 161
.lms, 206
.lof, 158
.lsm, 209
.map, 168
.mdb, 209
.mea, 130
.mnc, 166
.mng, 167
.mod, 142
.mov, 189
.mrc, 168
.mrcs, 168
.mrw, 166
.msr, 145, 156
.mtb, 173
.mvd2, 204
.naf, 135
.nd, 162
.nd2, 171
.ndpi, 136
.ndpis, 136
.nef, 169
.nhdr, 172
.nii, 170
.nii.gz, 170
.nrrd, 172
.obf, 145
.obsep, 173
.oib, 173
.oif, 173
.oir, 175
.ome, 178
.ome.btf, 177
.ome.tf2, 177
.ome.tf8, 177
.ome.tif, 177
.ome.tiff, 177
.ome.xml, 178
.omp2info, 176
.par, 150
.pbm, 184
.pcoraw, 180
.pcx, 181
.pds, 181
.pgm, 184
.pic, 119
.pict, 186
.png, 112, 187
.pnl, 114
.ppm, 184
.pr3, 199
.ps, 129
.psd, 185
.qptiff, 201
.r3d, 127
.raw, 119, 143, 172
.rcpnl, 127
.rec, 168, 180
.res, 130
.scn, 120, 159, 161
.sdt, 117
.seq, 140
.sif, 111
.sld, 105, 199
.sldy, 106
.sm2, 190
.sm3, 190
.spc, 117
.spe, 188
.spi, 194
.st, 168
.stk, 162, 194
.stp, 167
.svs, 112, 113
.sxm, 160
.tf2, 196
.tf8, 196
.tfcyto, 198
.tfr, 198
.tga, 194
.tif, 107, 116, 125, 126, 138, 144, 145, 147, 150, 156, 160, 161, 163–165, 169, 173, 174, 176, 182, 183, 185, 188, 195, 196, 199, 201, 207
.tiff, 131, 147, 160, 162, 163, 165, 170, 171, 177, 183, 185, 193, 196
.tnb, 173
.top, 179
.txm, 210
.txrm, 210
.txt, 138, 164, 172, 195
.v, 129
.vff, 134
.vms, 137
.vsi, 125
.vws, 197
.wat, 205
.wpi, 126
.xdce, 145
.xlef, 159
.xml, 118, 119, 125, 164, 176, 182, 183, 188, 207
.xqd, 191
.xqf, 191
.xv, 154
.xys, 203

.zfp, 198
.zfr, 198
.zvi, 208
-
    showinf command line option, 25
-autoscale
    showinf command line option, 24
-bigtiff
    bfconvert command line option, 28
-cache
    bfconvert command line option, 29
    showinf command line option, 24
-cache-dir
    bfconvert command line option, 29
    showinf command line option, 24
-channel
    bfconvert command line option, 26
-columns
    mkfake command line option, 34
-compression
    bfconvert command line option, 27
-crop
    bfconvert command line option, 27
    showinf command line option, 23
-debug
    mkfake command line option, 34
    showinf command line option, 24
-extra-metadata
    bfconvert command line option, 30
-fast
    showinf command line option, 24
-fields
    mkfake command line option, 34
-fill
    bfconvert command line option, 29
    showinf command line option, 25
-format
    showinf command line option, 25
-help
    formatlist command line option, 33
-html
    formatlist command line option, 33
-no-sas
    bfconvert command line option, 29
-no-sequential
    bfconvert command line option, 29
-no-upgrade
    showinf command line option, 23
-nobigtiff
    bfconvert command line option, 28
-nocore
    showinf command line option, 24
-noflat
    bfconvert command line option, 25
    showinf command line option, 22
-nolookup
    bfconvert command line option, 28
-nooverwrite
    bfconvert command line option, 28
-nopix
    showinf command line option, 23
-novalid
    showinf command line option, 23
-omexml
    showinf command line option, 23
-omexml-only
    showinf command line option, 24
-option
    bfconvert command line option, 25
    showinf command line option, 23
-overwrite
    bfconvert command line option, 27
-padded
    bfconvert command line option, 28
-plates
    mkfake command line option, 34
-precompressed
    bfconvert command line option, 29
-pyramid-resolutions
    bfconvert command line option, 28
-pyramid-scale
    bfconvert command line option, 28
-quality
    bfconvert command line option, 27
-range
    bfconvert command line option, 26
    showinf command line option, 23
-rows
    mkfake command line option, 34
-runs
    mkfake command line option, 34
-series
    bfconvert command line option, 26
    showinf command line option, 22
-swap
    bfconvert command line option, 29
    showinf command line option, 24
-tilex
    bfconvert command line option, 26
-tiley
    bfconvert command line option, 26
-timepoint
    bfconvert command line option, 26
-txt
    formatlist command line option, 32
-xml
    formatlist command line option, 33
-z

bfconvert command line option, 26
3i SlideBook, 105
3i SlideBook 7, 106

A

Adobe Photoshop PSD, 185
AIM, 108
Alicona 3D, 108
Amersham Biosciences Gel, 109
Amira Mesh, 110
Amnis FlowSight, 110
Analyze 7.5, 111
Andor Bio-Imaging Division (ABD) TIFF, 107
Andor SIF, 111
Animated PNG, 112
Aperio AFI, 112
Aperio SVS TIFF, 113
Applied Precision CellWorX, 114
AVI (*Audio Video Interleave*), 115
Axon Raw Format, 116

B

BD Pathway, 116
Becker & Hickl SPC FIFO, 117
Becker & Hickl SPCImage, 117
BF_CP, **21**, 21
BF_DEVEL, 22
BF_FLAGS, **21**, 21
BF_MAX_MEM, **21**, 21
BF_PROFILE, **21**, 21, 22
BF_PROFILE_DEPTH, **21**, 21
bfconvert, **21**
bfconvert command line option
    -bigtiff, 28
    -cache, 29
    -cache-dir, 29
    -channel, 26
    -compression, 27
    -crop, 27
    -extra-metadata, 30
    -fill, 29
    -no-sas, 29
    -no-sequential, 29
    -nobigtiff, 28
    -noflat, 25
    -nolookup, 28
    -nooverwrite, 28
    -option, 25
    -overwrite, 27
    -padded, 28
    -precompressed, 29
    -pyramid-resolutions, 28
    -pyramid-scale, 28
    -quality, 27
    -range, 26
    -series, 26
    -swap, 29
    -tilex, 26
    -tiley, 26
    -timepoint, 26
    -z, 26
Big Data Viewer, 118
Bio-Rad Gel, 118
Bio-Rad PIC, 119
Bio-Rad SCN, 120
Bitplane Imaris, 121
Bruker MRI, 122
BSD, **105**
Burleigh, 122

C

Canon DNG, 123
CellH5, 124
Cellomics, 124
cellSens VSI, 125
CellVoyager, 125
CLASSPATH, 48, 55, 58, 59, 62, 63, 81
CV7000, 126

D

DeltaVision, 127
DICOM, 127
domainlist, **21**

E

ECAT7, 129
environment variable
    BF_CP, 21
    BF_DEVEL, 22
    BF_FLAGS, 21
    BF_MAX_MEM, 21
    BF_PROFILE, 21, 22
    BF_PROFILE_DEPTH, 21
    CLASSPATH, 48, 55, 58, 59, 62, 63, 81
EPS (*Encapsulated PostScript*), 129
Evotec/PerkinElmer Opera Flex, 130
Export, **105**

F

FEI, 130
FEI TIFF, 131
FITS (*Flexible Image Transport System*), 132
formatlist, **21**
formatlist command line option
    -help, 33
    -html, 33
    -txt, 32

-xml, 33

G

Gatan Digital Micrograph, 132
Gatan Digital Micrograph 2, 133
GE MicroCT, 134
GIF (*Graphics Interchange Format*), 134

H

Hamamatsu Aquacosmos NAF, 135
Hamamatsu DCIMG, 135
Hamamatsu HIS, 136
Hamamatsu ndpi, 136
Hamamatsu VMS, 137
Hitachi S-4800, 138

I

I2I, 138
ICS (*Image Cytometry Standard*), 139
ijview, **21**
Imacon, 140
ImagePro Sequence, 140
ImagePro Workspace, 141
IMAGIC, 141
IMOD, 142
Improvision Openlab LIFF, 143
Improvision Openlab Raw, 143
Improvision TIFF, 144
Imspector OBF, 145
InCell 1000/2000, 145
InCell 3000, 146
INR, 146
Inveon, 147
Ionpath MIBI, 147
IPLab, 148
IVision, 149

J

JDCE, 150
JEOL, 150
JPEG, 151
JPEG 2000, 152
JPK, 152
JPX, 153

K

Keller Lab Block, 153
Khoros VIFF (Visualization Image File Format) Bitmap, 154
Kodak BIP, 154

L

Lambert Instruments FLIM, 155
LaVision Imspector, 156
Leica LAS AF LIF (*Leica Image File Format*), 157
Leica LCS LEI, 156
Leica LOF, 158
Leica SCN, 159
Leica XLEF, 159
LEO, 160
Li-Cor L2D, 161
LIM (*Laboratory Imaging/Nikon*), 161

M

Metadata, **105**
MetaMorph 7.5 TIFF, 162
MetaMorph Stack (*STK*), 162
MetaXpress, 163
MIAS (*Maia Scientific*), 164
Micro-Manager, 164
Mikroscan TIFF, 165
MINC MRI, 166
Minolta MRW, 166
mkfake, **21**
mkfake command line option
-columns, 34
-debug, 34
-fields, 34
-plates, 34
-rows, 34
-runs, 34
MNG (*Multiple-image Network Graphics*), 167
Molecular Imaging, 167
MRC (*Medical Research Council*), 168
Multiple Images, **105**

N

NEF (*Nikon Electronic Format*), 169
NIfTI, 170
Nikon Elements TIFF, 170
Nikon EZ-C1 TIFF, 171
Nikon NIS-Elements ND2, 171
NRRD (*Nearly Raw Raster Data*), 172

O

Olympus CellR/APL, 173
Olympus FluoView FV1000, 173
Olympus FluoView TIFF, 174
Olympus OIR, 175
Olympus OMP2INFO, 176
Olympus ScanR, 176
Olympus SIS TIFF, 177
OME-TIFF, 177
OME-XML, 178
OMERO Pyramid, 179
Openness, **105**
Oxford Instruments, 179

P

PCORAW, 180
PCX (*PC Paintbrush*), 181
Perkin Elmer Densitometer, 181
PerkinElmer Columbus, 182
PerkinElmer Nuance, 182
PerkinElmer Operetta, 183
PerkinElmer UltraVIEW, 183
Photoshop TIFF, 185
PicoQuant Bin, 186
PICT (*Macintosh Picture*), 186
Pixels, **105**
PNG (*Portable Network Graphics*), 187
Portable Any Map, 184
Prairie Technologies TIFF, 188
Presence, **105**
Princeton Instruments SPE, 188
Pyramid, **105**

Q

Quesant, 189
QuickTime Movie, 189

R

Ratings legend and definitions, **105**
RHK, 190

S

SBIG, 191
Seiko, 191
showinf, **20**
showinf command line option
    -, 25
    -autoscale, 24
    -cache, 24
    -cache-dir, 24
    -crop, 23
    -debug, 24
    -fast, 24
    -fill, 25
    -format, 25
    -no-upgrade, 23
    -nocore, 24
    -noflat, 22
    -nopix, 23
    -novalid, 23
    -omexml, 23
    -omexml-only, 24
    -option, 23
    -range, 23
    -series, 22
    -swap, 24
SimplePCI & HCImage, 192
SimplePCI & HCImage TIFF, 193
SM Camera, 193
SPIDER, 194

T

Targa, 194
Tecan Spark Cyto Workspace, 195
Text, 195
TIFF (*Tagged Image File Format*), 196
tiffcomment, **21**
TillPhotonics TillVision, 197
TissueGnostics TFCYTO, 198
Topometrix, 198
Trestle, 199

U

UBM, 199
Unisoku, 200
Utility, **105**

V

Varian FDF, 200
Vectra QPTIFF, 201
Veeco AFM, 202
Ventana BIF, 202
VG SAM, 203
VisiTech XYS, 203
Volocity, 204
Volocity Library Clipping, 204

W

WA-TOP, 205
Windows Bitmap, 206

X

xmlindent, **21**
xmlvalid, **21**

Z

Zeiss Axio CSM, 206
Zeiss AxioVision TIFF, 207
Zeiss AxioVision ZVI (*Zeiss Vision Image*), 208
Zeiss CZI, 209
Zeiss LSM (Laser Scanning Microscope) 510/710, 209
Zeiss XRM, 210