SOLUTIONS FOR CONVERTING IMAGES INTO NEXT-GENERATION FILE FORMATS (NGFF)

A report written in fulfilment of deliverable D3.2 of the MSCA-AGILE project

Sanat Sanjay Chitnis
Université Clermont Auvergne
Supervised by Prof. Dr. Christophe Tatout, IRP2

Anna Beatriz de Souza Perotto
University of Dundee
Supervised by Dr. Jean-Marie Burel, IRP13

Genevieve Laprade
University of Zürich
Supervised by Prof. Dr. Virginie Uhlmann, IRP15

October 2026

Summary. This report provides solutions to converting images into the Next-Generation File Format (NGFF) OME-Zarr. OME-Zarr is an open, chunked, multi-resolution format for large images, and open-source tools facilitate conversion from other formats. OME-Zarr is becoming the preferred format to store images in public repository such as Bioimage Archive (BIA) or Image Data Resource (IDR) and is poised to become a standard for microscopy data management and analysis. This report establishes the context for the file format, presents options for conversion tools, and walks readers through the steps for using them. Limitations and further learning resources are also addressed.

Version 1.0

Table of Contents

Introduction *3*
Common imaging file formats **3**
Zarr **3**
OME-Zarr **4**
The need for converter tools **4**
Converter Tools *5*
List of convertor tools: **5**
Guidelines *6*
Limitations *8*
Not yet at version 1.0 ..... **8**
Minimal documentation for beginners **8**
Conversion from proprietary file formats **8**
Storage Cost **8**
Technical expertise for OMERO interoperability **8**
Conclusion *9*
Learning Resources *10*
Glossary *11*
References *12*

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

Introduction

The field of bioimaging has rapidly evolved in recent years, leading to the development of new imaging modalities and analysis tools. Although these developments have provided new directions in scientific research, they have also resulted in a growing number of proprietary file formats[1]. Consequently, rather than adopting a unified format, an ever-increasing number of dedicated converters are being developed and consolidated within the Bio-Formats library. This highlights the importance of making imaging data adhere to the FAIR (Findable, Accessible, Interoperable and Reusable) principles.[2] To improve the interoperability and accessibility of proprietary file formats, open-source libraries have been developed to read and convert images from different formats into common data representations.[1]

Common imaging file formats

Formats such as TIFF and HDF5 are popular choices for conversion. TIFF is widely used for 2D images (dimensions X, Y), but can be less efficient for large multidimensional datasets, such as 3D-5D images with dimensions representing time-points (T), channels (C), and spatial coordinates (Z, Y, X). These datasets may be stored across multiple files or as multipage TIFFs, where each page represents a 2D plane. If the user only requires a small region of the 3D area, accessing the required region may involve reading entire 2D planes, which results in loading unnecessary amounts of data into memory. [1,3]

In contrast, HDF5 supports multidimensional arrays and allows data to be organised into smaller blocks called chunks, which allows selected portions to be accessed without loading the entire dataset into memory. For example, in a 3D image of 50 slices where (Z, Y, X) = (50, 2048, 2048) a chunk could be the first slice with (Z, Y, X) = (1, 2048, 2048).

Both TIFF and HDF5 formats are useful for local machine usage but can present challenges when used in large-scale cloud-based object storage environments. In multipage TIFF files, accessing a small region of an image may require reading larger portions of relevant 2D planes. HDF5 allows chunking but the chunks are stored in a single file. This makes it less suitable for cloud storage and retrieval, as accessing data requires requests over a network. The data loading software must locate the required chunks from a single file which causes delay and leads to slow data loading. [1,3]

Zarr

Zarr is a data storage format designed to efficiently store and access large multidimensional arrays, particularly in cloud-based environments. Like HDF5, Zarr divides the image data is into chunks, which can be compressed and accessed individually. However, unlike HDF5 storage, Zarr stores chunks as individual objects rather than in a single file. This allows the software that is loading the data to locate and retrieve only chunks required for a specific image region rather than locating them in a single large file. This structure also allows multiple chunks to be retrieved in parallel, which reduces data loading time. Zarr also stores metadata such as the dimensions (e.g. Z, Y, X = 50, 2048, 2048), data type representing pixel intensities (e.g. integer types from 8bit to 64 bit), chunk configuration (a slice with Z, Y, X = 1, 2048, 2048)

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

and compression algorithm (e.g. Blosc). Zarr also stores data in a container that follows a hierarchical structure similar to a computer's folder tree: the data is stored in groups (equivalent to folders) that can contain subgroups, multidimensional arrays, and their associated metadata. These features make Zarr suitable for storing, accessing, and sharing large multidimensional bioimages in cloud-based environments, attributes that are not optimised in HDF5[4].

OME-Zarr

While Zarr offers a generic way to store multidimensional arrays, divide them into chunks, compress the data, and organise groups and subgroups, it does not define standardised conventions for describing microscopy specific information. The Open Microscopy Environment (OME) consortium developed the OME Model to provide a common specification for describing biological imaging data, including image dimensions, pixel types, acquisition details, annotations, and regions of interest (ROIs)[5]. This model helps in the interpretation of imaging data across different software varieties and is used in formats such as OME-XML and OME-TIFF.

Building on this approach, OME-Zarr extends Zarr with conventions for organising multidimensional image and their associated metadata. The OME-Zarr 0.6 specification uses Zarr v3 and defines metadata for image axes, coordinate systems, physical scales, and multiscale image data[6]. OME-Zarr supports a pyramidal structure, also known as a multiscale image, in which an image is stored at multiple resolutions. OME-Zarr constructs a multiscale image by storing both the original full-resolution image and the progressively downsampled versions as Zarr arrays. This allows, among many other applications, viewer applications to display larger-than-memory image initially at low resolution, while zooming in retrieves higher resolution data for finer details. This reduces the amount of data that needs to be accessed at any one time[3,6].

The need for converter tools

Bioimaging data continues to be generated in a wide range of formats, and current popular file formats are either proprietary or not optimised for large scale, cloud-based storage. To benefit from OME-Zarr's capabilities, existing datasets need to be converted to the OME-Zarr format. Converting to a standardised file format eliminates the need for repeated file conversions and allows data and metadata to be stored in a standardised structure that is independent of the original acquisition system[1,3]. Therefore, converter tools are essential to helping transition from multiple formats into OME-Zarr. The following sections highlight the currently available tools that can be used to convert other bioimaging file formats to OME-Zarr.

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

Converter Tools

When selecting a converter tool, the most important factor is your working environment and how you prefer to operate it. Consider whether you want to use a graphical user interface (GUI), run it from a command line interface (CLI), run it in batch mode on a High-Performance Computing (HPC) cluster, or a plugin integrated into a software you already use such as Fiji or Napari. The OME-Zarr community keeps an up-to-date list of converters online at NGFF-Zarr Converters.[7] Below, we list brief descriptions of each tool and the circumstances in which you would use them. Brief descriptions of each tool and the circumstances in which you would use them.

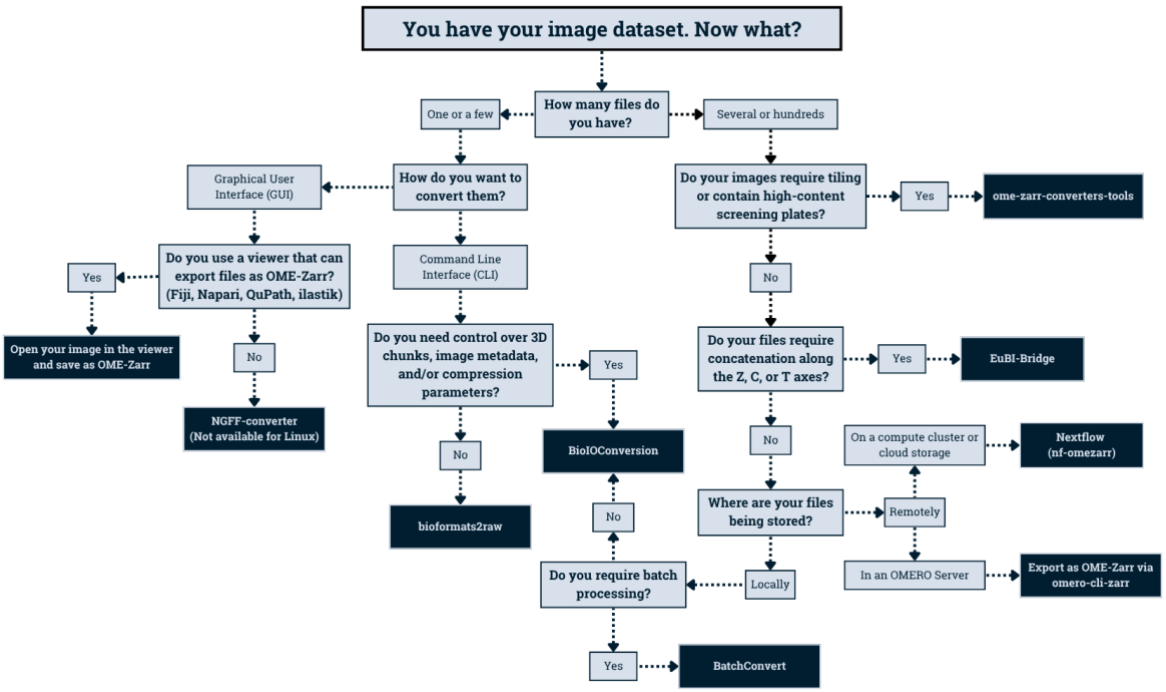

Figure 1: ***Choosing a converter.*** *Decision tree for the converter tool, starting from the number of files. For one or a few files the choice is between a graphical interface (1. An image viewer or 2. NGFF-Converter) and the command line interface (3. bioformats2raw, or 4. BioIOConversion). For many files, first decide if they require tiling or contain high-content screening plates (5. ome-zarr-converters-tools), then if they must be concatenated into one image (6. EuBI-Bridge), then where the data lives: on a cluster or in cloud storage (7. nf-omezarr), in an OMERO server (8. omero-cli-zarr export), or locally. If local files need batch conversion, then use 9. BatchConvert, otherwise use 4. BioIOConversion. Note that this flowchart is simply a guideline for choosing a tool based on the files you have and how you want to process them; many converters can be used to execute the same operations.*

List of convertor tools:

1. Viewers that can export image file as OME-Zarr:

1.1. Fiji[9]: Using the n5-ij plugin, users can export an open image as an OME-Zarr file using the *File → Save As → Export HDF5/N5/Zarr/OME-NGFF* option.

1.2. Napari[10]: Users can export to OME-Zarr using the ome-zarr-py Python library in Napari's console.

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

1.3. QuPath[11]: QuPath provides a built-in OME-Zarr export option under *File → Export images → OME-Zarr*.

1.4. ilastik[12]: This tool provides multiscale exports by generating progressively downsampled images levels. Export to AWS S3 is experimentally supported but untested.

2. NGFF-Converter: This is a graphical application which packages bioformats2raw, simplifying conversion for users less familiar with CLI tools. This application supports customisable conversion settings and batch processing, which allows users to schedule multiple conversion jobs and monitor their progress.

3. bioformats2raw: This is the original Java-based CLI tool for converting file formats supported by Bio-Formats into OME-Zarr. It allows the user to customise conversion settings including chunk size, compression, downsampling, and the number of resolution levels. It is well suited to automated workflows, batch conversion, and HPC environments.

4. bioio-conversion: This is a Python-based convertor tool which accommodates both single file and batch conversion. The OMEZarrConverter module within bioio supports multiscale pyramid generation and allows users to customise chunk sizes, compression settings, and image metadata, including channel information, axis labels, and physical pixel size.

5. ome-zarr-converters-tools: This is a Python library developed by the BioVisionCenter that provides shared utilities for building custom OME-Zarr converters. It supports filtering and validating image tiles, combining them into complete images, and correcting tile positions to create stitched mosaics. The library can generate OME-Zarr datasets from standalone images or high-content screening (HCS) plate layouts.

6. EuBI-Bridge: This is a Python-based tool which converts microscopy image collections into OME-Zarr, with both CLI and GUI options. Its distinguishing feature is an aggregative conversion, which combines multiple input files along user-specified dimensions, such as channels and timepoints, to reconstruct multidimensional datasets from separate files.

7. nf-omezarr: This is a Nextflow pipeline developed by researchers at HHMI Janelia to automate the conversion of Bio-Formats compatible images into OME-Zarr using Bioformats2raw. This tool also supports batch processing of image directories and is designed for HPC environments. This pipeline supports containerised dependencies.

8. omero-cli-zarr: This is a CLI plugin that supports importing OME-Zarr data into OMERO and exporting OMERO images to OME-Zarr. Import needs the omero-zarr-pixel-buffer extension on the server.

9. BatchConvert: This is a CLI tool developed by Euro-Bioimaging. It also uses Bioformats2raw to perform conversions in addition to Nextflow[8] to support parallel processing of image batches. This tool also supports local and remote storage, including S3 compatible object storage, and can run on HPC clusters using Slurm.

Guidelines

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

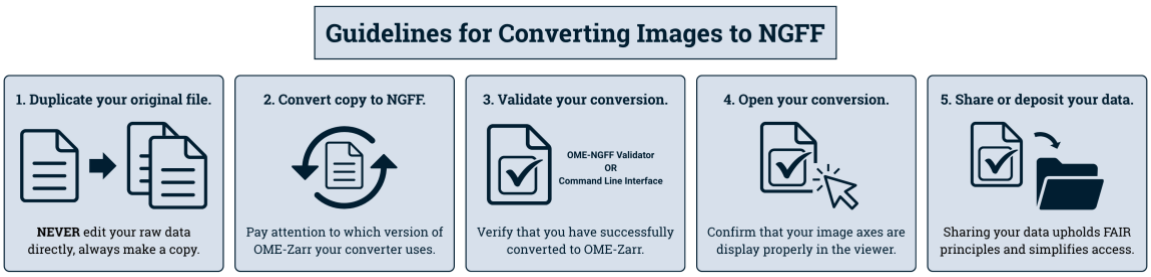

Figure 2: Guidelines for converting images to NGFF

1. **Make a copy of your original file**
   Never edit your raw data directly, always duplicate it. If the conversion fails, the original file is the only complete record of the acquisition. Storing this original copy allows you to make as many duplicates as you need while keeping the raw data intact. During the conversion process, Bio-Formats maps only the metadata fields it knows for each format, and OME-Zarr carries the rest of the metadata into OME/METADATA.ome.xml, which is a side file that only a few readers use.
2. **Convert your image to NGFF**
   First convert using the default settings of the converter tool or choose a version deliberately. Most tools write OME-Zarr v0.4 (Zarr v2), which is interoperable with most viewers. Depending on the converter you use, you may be able to control chunk size, compression, and pyramidal parameters.
3. **Validate that your conversion was successful**
   Use OME-NGFF validator and/or follow the guidelines in the ome-zarr-models documentation to ensure that you have produced a valid file. OME-NGFF validator can also be used to fulfil step 4. To save space, delete your working copy once the output is validated.
4. **Open your conversion to confirm that the view looks correct**
   Confirm that your Z, C, and/or T axes display properly in at least one viewer.
   The community maintains a list of OME-Zarr viewers that you can use to check your result image: Zarr-Viewers
5. **Share or deposit your NGFF file**
   To facilitate sharing your data with collaborators, you can upload OME-Zarr as a zipped file to the BioImage Archive (BIA) or Image Data Resource (IDR).

If you encounter problems with the conversion process and are unable to resolve them with the documentation alone, the image.sc forum provides a place to ask your questions and troubleshoot with expert users.

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

Limitations

Not yet at version 1.0

OME-Zarr has not yet reached a stable 1.0 release. As of the construction of this report, the latest version is 0.6. Future changes that are desired prior to the 1.0 release are specified under the "Request for Comments" (RFCs) section of the OME-Zarr documentation on the NGFF website[7]. Current issues being addressed by the RFCs include constraints on axis properties, lack of ability to specify spatial axis orientation, and limited transformation capabilities. Once addressed, these limitations will allow OME-Zarr to have increased longevity with flexible metadata interoperability. Detailed descriptions can be found on the aforementioned RFCs page.

Minimal documentation for beginners

While the existing documentation is extremely thorough for developers, it may be technically complicated for non-computational scientists. As the 1.0 release has not been reached yet, documentation that is beginner-friendly and technically sound remains limited.

Conversion from proprietary file formats

Since microscope manufacturers typically package images in file formats that are specific to their private software, converting to OME-Zarr may result in loss of incompatible metadata[2,3].

Storage Cost

Conversion multiplies storage. During conversion, you hold three files: the original, the copy and the OME-Zarr. One workaround to this is deleting the working copy once the OME-Zarr output has been validated.

Technical expertise for OMERO interoperability

While it is possible to work with OME-Zarr in OMERO[13], its current state is not broadly accessible to users of different skill levels. Working with this file format in OMERO requires specific plugins and CLI commands[14]. The omero-zarr-pixel-buffer extension allows an OMERO server to read an OME-Zarr file locally or remotely, without copying it, and serves its pixels as if the image had been imported normally[14]. Users who are not experienced with these operations may encounter difficulty when attempting to use them.

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

Conclusion

OME-Zarr presents a viable solution to the task of data standardisation in the field of microscopy through its continual maintenance and adhesion to FAIR principles. Here, we have summarised the current status of the OME-Zarr file format and its most recent usage tools. While it has not yet reached its 1.0 release, OME-Zarr is an adaptive and utile tool for storing and working with large image data. While TIFF and HDF5 are commonly used formats for working with datasets of these magnitudes, they are not optimised for cloud storage or efficient memory usage. The Zarr file format overcomes these storage limitations, and OME-Zarr adapts Zarr to microscopy metadata. The active OME-Zarr community and its ongoing maintenance set a precedent for how data standards should be curated: with long-term sustainability in mind rather than a short-term solution. Prospective users of OME-Zarr should use these resources to apply this tool to their work. Further limitations are still preventing widespread use; however, several RFCs are undergoing work to expand upon the interoperability of NGFFs. Collaborations with private companies to ensure that proprietary formats and OME-Zarr can coexist and be used in an interoperable way are desired and mutually beneficial. The learning resources below are a good place to start your OME-Zarr journey. Please note that this report reflects the state of available tools at the time it was written. Information is subject to change as development progresses, so refer to the latest documentation for the best results.

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

Learning Resources

NGFF Learning and Teaching Material:

A comprehensive list provided by OME listing readings, tutorials, and videos for learning how to work with NGFF.

NeuBIAS OME-Zarr tutorial:

Hands-on tutorials with BatchConvert, FIJI, Napari and the validator.

An Introduction to OME-Zarr for Big BioImage Data:

A book explaining the theory behind OME-Zarr and providing context for its usage.

OME-Zarr for OMERO users:

A guide for working with OME-Zarr file formats in OMERO.

Image.sc forum:

A forum for posting imaging and image analysis questions of all types and receiving help from expert users.

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

Glossary

Amazon Simple Storage Service (S3): Amazon's cloud object storage. Objects are stored as named units in a flat space, not in folders, and read over the web in parallel.

Bio-Formats: the open-source Java library maintained by OME that reads more than 150 microscopy file formats into the OME Model; most converters use it to read vendor files.

BioImage Archive (BIA): EMBL-EBI's public repository for bioimaging data. BIA accepts OME-Zarr datasets as submission files.

Chunk: a small block of an image stored as its own compressed file, so that each chunk can be read, written and processed independently of the others.

CLI (command-line interface): The text interface of a program, used from a terminal (Terminal on macOS and Linux, PowerShell or Command Prompt on Windows).

Compression: A way of storing data in a smaller file size. Can be lossless (no data is lost) or lossy (data is lost during compression)

Containerised: run inside a container, an isolated environment that carries its own copy of every dependency.

Downsampling: reducing an image's resolution to produce a smaller copy; each level of the pyramid is a downsampled version of the one above.

Euro-Bioimaging: The European research infrastructure for biological and biomedical imaging.

GUI (graphical user interface): a program's visual front end, where users point and click rather than type commands.

HPC cluster: High-performance computing; shared computers that are connected to remotely on which many jobs run in parallel, usually scheduled with Slurm.

Image Data Resource (IDR): public repository of reference microscopy datasets

Metadata: information about an image other than its pixel values: pixel size, axes, channel names, acquisition settings.

Nextflow: workflow software that runs many jobs in parallel on a laptop, a cluster or the cloud.

OMERO: OME's server platform for storing, annotating and sharing microscopy data.

Pyramid (Multiscale Image): A hierarchical way of storing images so that they can easily be scaled to different resolution levels without requiring that all data be loaded at once. Google Maps is often used as an analogy, with its ability to zoom in to different scales of view.

RFC: Request for comments; the OME-Zarr format for processing potential version updates

Slurm: software for scheduling jobs on most HPC clusters. BatchConvert and nf-omezarr submit conversions to it so that many files convert in parallel.

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)

References

1. Moore, J., Allan, C., Besson, S. *et al.* OME-NGFF: a next-generation file format for expanding bioimaging data-access strategies. *Nat Methods* **18**, 1496–1498 (2021). https://doi.org/10.1038/s41592-021-01326-w
2. Wilkinson, M., Dumontier, M., Aalbersberg, I. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* **3**, 160018 (2016). https://doi.org/10.1038/sdata.2016.18
3. Moore, J., Basurto-Lozada, D., Besson, S. *et al.* OME-Zarr: a cloud-optimized bioimaging file format with international community support. *Histochem Cell Biol* **160**, 223–251 (2023). https://doi.org/10.1007/s00418-023-02209-1
4. Miles, A., Striebel, J., Rzepka, N., Maitin-Shepard, J., and Moore, J. Zarr core specification. https://zarr-specs.readthedocs.io/en/v3.1.0/v3/core/index.html (2023)
5. Linkert, M., Rueden, C., Allan, C., *et al.* Metadata matters: access to image data in the real world. *J Cell Biol* **189 (5)**, 777-782. (2010) https://doi.org/10.1083/jcb.201004104
6. Moore, J., Bogovic, J., Soltwedel, J. *et al.* Version 0.6. https://ngff.openmicroscopy.org/specifications/0.6/index.html (2025)
7. OME-Zarr. https://ngff.openmicroscopy.org/index.html (2025)
8. Di Tommaso, P., Chatzou, M., Floden, E. et al. Nextflow enables reproducible computational workflows. Nat Biotechnol 35, 316–319 (2017). https://doi.org/10.1038/nbt.3820
9. Schindelin, J., Arganda-Carreras, I., Frise, E. *et al.* Fiji: an open-source platform for biological-image analysis. Nat Methods **9**, 676–682 (2012). https://doi.org/10.1038/nmeth.2019
10. Chiu, C.-L., Clack, N. & the napari community. napari: a Python Multi-Dimensional Image Viewer Platform for the Research Community. Microsc. Microanal. **28**, 1576–1577 (2022). https://doi.org/10.1017/S1431927622006328
11. Bankhead, P., Loughrey, M.B., Fernández, J.A. *et al.* QuPath: Open source software for digital pathology image analysis. *Sci Rep* **7**, 16878 (2017). https://doi.org/10.1038/s41598-017-17204-5
12. Berg, S., Kutra, D., Kroeger, T. et al. ilastik: interactive machine learning for (bio)image analysis. *Nat Methods* 16, 1226–1232 (2019). https://doi.org/10.1038/s41592-019-0582-9
13. Lubina, T. Investigating OMERO and OME-Zarr support – Overview as of July 2026. *image.sc.* https://forum.image.sc/t/investigating-omero-and-ome-zarr-support-overview-as-of-july-2026/121787 (2026)
14. Allan, C., Burel, JM., Moore, J. *et al.* OMERO: flexible, model-driven data management for experimental biology. Nat Methods **9**, 245–253 (2012). https://doi.org/10.1038/nmeth.1896

This project has been funded by the European Union under Horizon Europe (Grant Agreement n°101226536)