The Evolution of Scientific Data Formats

Learnings from ZIP to Zarr on New Standards for Scientific Publishing

Rowan Cockett[1,2]

[1]Continuous Science Foundation, [2]Curvenote Inc.

Abstract

We motivate an Open Exchange Architecture (oxa) for scientific publishing, informed by the progression from ZIP archives to HDF5 stores to Zarr-based object collections. By abstracting articles, data, code, and review materials into addressable, typed components, we want an open exchange architecture that can preserve the semantic richness of JATS while matching the access patterns of modern cloud-native science. We describe some of the interoperability requirements needed to make scholarly communication as streamable, reproducible, and machine-actionable as contemporary scientific datasets.

In the world of scientific publishing, new standards are often met with a groan, followed by someone sharing that xkcd comic about "there being 14 competing standards… now 15." It's funny because it's true — ***sometimes***. But other times, it misses the point entirely. Sometimes, a new standard isn't about adding noise — it's about enabling something fundamentally **new**.

The evolution from **ZIP arrow.r HDF5 arrow.r Zarr** tells that story.

1. The ZIP Era — Sharing Files

In 1989, Phil Katz introduced the **ZIP** format, a simple and open way to compress and bundle multiple files. ZIP solved a *distribution* problem: getting data from one machine to another efficiently. It standardized something broad — the ability to share complex folders across systems — and quickly became a cornerstone of digital exchange.

But ZIP had limits. It was built for *files*, not *data*. It couldn't efficiently handle massive arrays or structured scientific content. It was a transport format, not a computational one.

Capability unlocked file portability.
Ecosystem enabled software distribution, document sharing.
Limitation flat archives; no structure or metadata for science.

2. The HDF5 Era — Storing and Structuring Science

By the late 1990s, data volumes exploded. Scientific simulations, satellite observations, and sensor networks needed something more powerful. Enter **HDF5** (1998), born at NCSA and later maintained by The HDF Group.

HDF5 introduced a **hierarchical, self-describing** data model — groups, datasets, attributes — built to manage large, structured, multidimensional arrays. It supported parallel I/O, chunking, and compression — critical in high-performance computing.

Continuous Science Foundation

Published Nov 09, 2025

Correspondence to
Rowan Cockett
rowan@continuousfoundation.org

HDF5 didn't replace ZIP; it **transcended it**. It addressed a new challenge: *efficient access to scientific structure at scale*. Entire disciplines built on it — from NASA's Earth Observing System to climate modeling to particle physics.

Capability unlocked hierarchical data, parallel I/O.
Ecosystem enabled HPC and scientific data management.
Limitation local file systems; monolithic files don't scale to cloud object stores.

3. The Zarr Era — Cloud-Native, Chunked, and Distributed

Fast-forward to 2015. The world had moved to the cloud, and computation had gone distributed. **Zarr**, created by Alistair Miles, took inspiration from HDF5 but **reimagined** it for a new environment — object stores, HTTP access, parallel cloud computing, and modular data ecosystems.

Zarr stores each array chunk as a separate object (or file) and uses simple JSON metadata. That's it. This design makes Zarr "cloud-native": you can open, stream, and process pieces of a terabyte-scale dataset directly from S3 or GCS without downloading the whole thing.

It's not "HDF5, but in folders" — it's **a rethinking of what a scientific data container means when storage, compute, and collaboration all happen across the network**.

Capability unlocked distributed, parallel, object-store access.
Ecosystem enabled FAIR data, Pangeo, OME-Zarr, cloud-scale machine learning, and analysis pipelines.
Limitation evolving specs and community governance — still in progress, but essential for the future.

4. What This Evolution Shows

Each format wasn't just a "new standard". Each **reflected a transformation in infrastructure and possibility**:

When technology shifts — from disks to clusters to clouds — our standards must evolve with it. ZIP could never have supported petabyte-scale simulation data. HDF5 has challenges to fully adapt to cloud-native workflows [1], as well as embracing the simplicity of the new cloud buckets [2]. And that's the lesson: *new ecosystems often require new standards.*

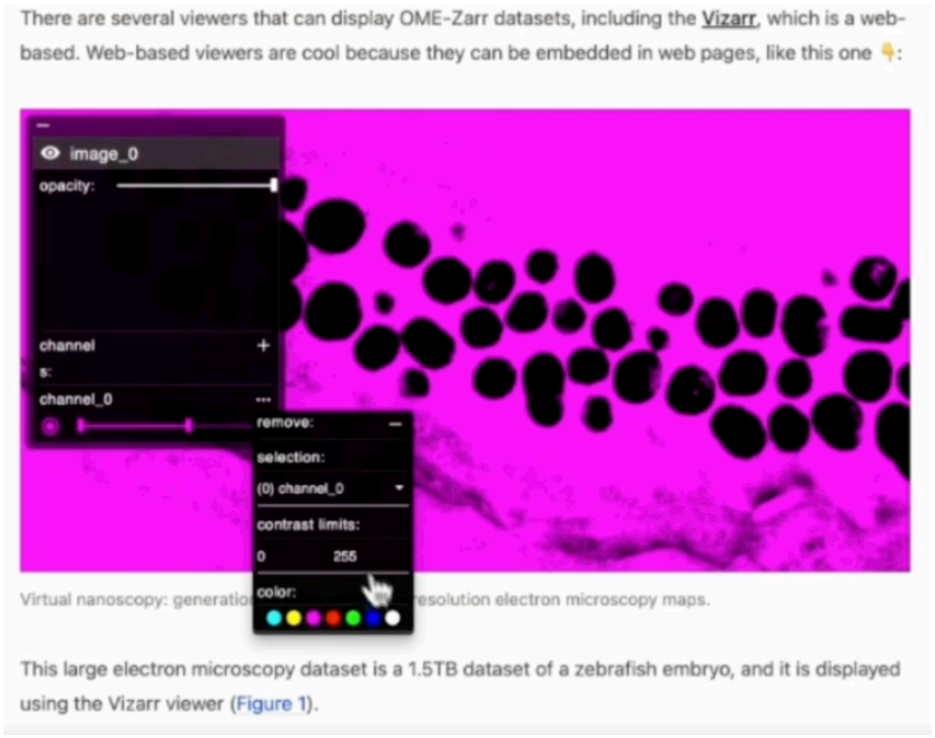
---

5. The Scientific Publishing Parallel — JATS and MECA

Around the same time HDF5 was taking root in scientific computing, publishing found its own "standard of record": **JATS** (the Journal Article Tag Suite), formalized in 2012 and updated in 2015 and 2021. JATS offered an XML vocabulary for describing every part of a scientific article — section headings, references, tables, figures, metadata. It has become a common export types of publishers and repositories, built for interoperability, preservation, and cross-platform rendering.

Table 1. *A timeline of data formats and the new scientific capabilities each unlocked*

Era	Format	New Capability	Why It Was Needed
1989	ZIP	Share files and folders	Cross-platform portability
1998	HDF5	Structure and access large scientific arrays	HPC & simulation workloads
2016	Zarr	Cloud-native chunked storage	Distributed and object-store compute



There are several viewers that can display OME-Zarr datasets, including the Vizarr, which is a web-based. Web-based viewers are cool because they can be embedded in web pages, like this one 👇:

Virtual nanoscopy: generatio… …esolution electron microscopy maps.

This large electron microscopy dataset is a 1.5TB dataset of a zebrafish embryo, and it is displayed using the Vizarr viewer (Figure 1).

Figure 1. *A 1.5TB zebrafish microscopy image using OME-Zarr. Under file-based standards or in print-based publishing, only a static PNG or small excerpt is directly viewable; the full-resolution microscopy data remain locked away, inaccessible for browsing or compute unless fully downloaded and extracted. In a cloud-enabled future, as we zoom in, the limitations of current publishing containers become clear: they lack the addressability and streaming needed for modern data-intensive science. These formats need to be accessible from the reading experience as well.*

To move these articles between systems, publishers introduced **MECA** — the Manuscript Exchange Common Approach — which packages an entire submission (the JATS XML, figures, supplementary materials) inside a **ZIP archive**, usually transferred via **FTP**.

JATS defines the **structure**. MECA defines the **container**. Together, they have become open infrastructure for scholarly publishing.

But that infrastructure still assumes a *file-based world*. One based around XML, FTP, and ZIP — tools that are seldom used in modern web-development workflows.

- JATS is an XML file. No images or other assets, primarily designed for text-only data-mining.
- MECA is a ZIP file. It can contain any amount of supplementary materials.
- And many publishers still use FTP servers to move those files between systems.

They're reliable and (relatively) standardized[1]. But they're not accessible to developers or to the component pieces of research. They don't interoperate with the tools researchers actually use: notebooks, data repositories, cloud archives, and computational environments.

In a world of APIs, data portals, and reproducible workflows, the "currency" of publishing remains static and opaque.

[1]There are many challenges with JATS with many different flavours and variants between each publisher. MECA has relatively small adoption in the industry, and often when implemented doesn't always follow the specification.

6. What If Scientific Publishing Made the Zarr Leap?

Imagine applying the "**Zarr Leap**" to scholarly publishing.

Zarr didn't reinvent the data or the data structures — it restructured *access*. It made scientific content *addressable, parallel, and web-native*.

What if publishing did the same?

Instead of wrapping an article and its supplements in a ZIP, what if we exposed the **components of research** — the notebook, data, code, figures, article, and review — as structured, typed, linked pieces that could be accessed directly?

Researchers, machines, and readers could:

- Pull a single figure panel with its provenance.
- Open a Jupyter cell that generated a result.
- Stream the dataset behind a table or image.
- Traverse from the article's text to its executable environment.
- Reuse, remix, and cite components **individually** — with fidelity, licensing, and attribution.

It would be like replacing FTP with APIs, and replacing ZIP with a structured bucket of interconnected, queryable components.

We wouldn't lose the semantic logic of JATS — we could **wrap it in a more expressive, cloud-native container**, just as Zarr wrapped the logic of HDF5 in a new access model. JATS was designed for paper; before a world of Jupyter Notebooks, linked protocols, datasets, and interactive articles — so there is *also* a lot we can and should do to improve the content that can be expressed.

7. A Thought to Leave With

Science has already made this leap once — from ZIP to HDF5 to Zarr — each time unlocking a new era of access, scale, and collaboration. Publishing hasn't yet made its equivalent jump. We are still shipping our most valuable ideas around as ZIP files on FTP servers.

It's time to imagine what publishing looks like when *articles are as open, structured, and explorable as data itself*. An Open Exchange Architecture, `oxa`.

Not a new standard for the sake of it — but a new standard for what's possible in scientific communication.

References

[1] M. Rocklin, "HDF in the Cloud: Challenges and Solutions for Scientific Data." [Online]. Available: https://matthewrocklin.com/blog/work/2018/02/06/hdf-in-the-cloud

[2] R. Abernathey, "Step-by-Step Guide to Building a Big Data Portal." [Online]. Available: https://medium.com/pangeo/step-by-step-guide-to-building-a-big-data-portal-e262af1c2977