**Community Standard Justification: Zarr**

**TITLE: Zarr v2 Storage Specification**

**Contributor Name: Ryan Abernathey & Other Zarr Steering Council**

**Organization: Zarr Steering Council**

**Phone:**

**Email: rpa@ldeo.columbia.edu**

**DATE: 2020-05-30**

# Introduction

This document provides a justification to the OGC Technical Committee (TC) for consideration of *Zarr* as a Community standard. This justification, along with the submitted candidate Community standard, will form the basis for TC review and vote to approve the start of a Work Item as the first step in the Community standard process for this standard.

The submitters agree to abide by the TC Policies and Procedures and OGC Intellectual Property Rights Policy (<http://www.opengeospatial.org/ogc/policies)> during the processing of this submission.

Once approved, the Community Standard Work Item defined by this document is valid for six (6) months.

# Overview of proposed submission

*Summarize the proposed Community standard. In this summary, provide an overview of the geospatial interoperability requirements the proposed standard supports, the history of its development and use, and use cases.*

Multidimensional array data (a.k.a. N-dimensional arrays, ND-arrays, “tensors”) is ubiquitous in scientific research and engineering. Zarr is an open-source specification for the storage of ND-arrays and associated metadata. Zarr stores metadata using .json text files and array data as [optionally] compressed binary chunks. The Zarr specification details the contents and layout of these elements. Zarr can store data into any storage system that can be described as a key/value store. In a standard filesystem, the keys are filenames within a directory hierarchy, and the values are the file contents. In a cloud object store (e.g., Amazon S3), the keys are the object IDs and the values are the object data. This flexibility allows implementations to experiment with novel storage technologies while maintaining a uniform API for downstream libraries and users.

Zarr arose in genomics research in 2016. It was created by Alistair Miles of Oxford as a library optimized for massively parallel array analytics. It has since grown into a community project with a range of developers and users from fields such as genomics, bioimaging, astronomy, physics, quantitative finance, oceanography, atmospheric science, climate science, and geospatial imaging. Because it can represent very large array datasets in a simple, scalable way, and is compatible with cloud object storage, Zarr is an ideal format for analysis-ready geospatial data in the cloud. A prominent example is the [Google Cloud CMIP6 Public Dataset](https://cloud.google.com/blog/products/data-analytics/new-climate-model-data-now-google-public-datasets), which currently comprises over 400 TB of Zarr data. While Zarr is not inherently a geospatial-specific format, because of its rapid growth and adoption in geospatial and related fields, we are proposing it as an OGC community standard.

The Zarr project is open source and practices open development on GitHub. The project is governed by a [steering council](https://github.com/zarr-developers/governance/blob/master/GOVERNANCE.md). [NumFocus](https://numfocus.org/project/zarr) has been the fiscal and legal sponsor of Zarr since 2019. In 2020, Zarr received a Chan-Zuckerberg [Essential Open Source Software](https://chanzuckerberg.com/eoss/proposals/scalable-storage-of-tensor-data-for-scientific-computing/) grant.

# Relationship to other OGC standards

*State whether this proposed Community standard has any dependencies on OGC standards or is itself normatively referenced by an OGC standard and list those standards, as applicable.*

Zarr is a new, standalone standard, without dependencies on existing OGC standards. However, because of its generic nature, other OGC standards could easily be implemented on top of Zarr. For example:

* Zarr could be the foundation for coverage data that are made available to a Web Coverage Service, similar to the OGC CF-netCDF 3.0 encoding using GML Coverage Application Schema (OGC 14-100r2)
* A future release of the netCDF library will [incorporate zarr](https://www.unidata.ucar.edu/blogs/news/entry/netcdf-and-native-cloud-storage) as one possible base storage container format.
* WKT representation of coordinate reference systems could be placed in zarr metadata.

# Alignment with OGC Standards Baseline

*Describe where this proposed standard fits with respect to the existing OGC standards baseline and standards in development in the OGC and whether this proposed standard may compete with or enhance an existing OGC standard*.

As a generic storage format for multidimensional array data + metadata, Zarr will play a similar role to HDF5 within the OGC ecosystem. Many different types of data and metadata can be stored using Zarr, targeting a wide range of storage technologies (filesystems, databases, cloud object store). We anticipate Zarr to be particularly popular for cloud-native data storage and processing. Also, because Zarr can store arbitrarily large arrays, without constraints related to individual file sizes, it will likely prove useful for creating homogeneous analysis-ready datasets from many individual granules

Zarr has already been adopted by several OGC communities as a format for cloud-optimized, analysis ready geospatial data. Examples include:

* Climate Science: The C[MIP6 Google Cloud Public Dataset](https://console.cloud.google.com/marketplace/details/noaa-public/cmip6)
* Oceanography: The [ECCOv4r3 Ocean State Estimate](https://catalog.pangeo.io/browse/master/ocean/ECCOv4r3/)
* Atmospheric Science: [Global cloud-resolving aquaplanet simulations with the System for Atmospheric Modeling](https://catalog.pangeo.io/browse/master/atmosphere/sam_ngaqua_qobs_eqx_2d/)

# Evidence of implementation

Zarr began with a Python implementation in 2016. Around the same time, the [N5 project](https://github.com/saalfeldlab/n5) emerged with very similar goals. In 2018, recognizing considerable overlap between the projects in terms of goals and technical choices, the two projects began to merge, a process which should complete by the time the Zarr v3 spec is released. In the meantime, Zarr has been implemented in Java, C++ (via Z5), Julia, and JavaScript. An integration repo exists to verify compatibility between different implementations: <https://github.com/constantinpape/zarr_implementations>.

The following implementations use the proposed Community standard.

**Implementation name**: Zarr Python

**Date of most recent version**: v2.3.1, 2020-03-25

**Implementation description**: Original implementation of Zarr, now the reference implementation.   
**Implementation URL**: <https://github.com/zarr-developers/zarr-python>, <http://zarr.readthedocs.io/>

**Is implementation complete**? x **Yes** ☐ **No**

**Implementation name**: Z5

**Date of most recent version**: v2.0.5, 2020-02-16

**Implementation description**: C++ and Python wrapper for zarr and n5 file formats. Implements the file system specification of these formats.   
**Implementation URL**: <https://github.com/constantinpape/z5>, <https://z5.readthedocs.io/>

**Is implementation complete**? ☐ **Yes**  x **No**

No support for cloud storage yet.

**Implementation name**: JZarr

**Date of most recent version**: last commit 2020-04-07

**Implementation description**: Java implementation of Zarr.   
**Implementation URL**: <https://github.com/bcdev/jzarr>, <https://jzarr.readthedocs.io/>

**Is implementation complete**? **Yes**  x **No**

Boolean and char type are not yet supported. Zip file store not supported.

**Implementation name**: Zarr.jl

**Date of most recent version**: v0.4.4, 2020-04-23

**Implementation description**: Julia implementation of Zarr.   
**Implementation URL**: <https://github.com/meggart/Zarr.jl>, <https://meggart.github.io/Zarr.jl>

**Is implementation complete**? **Yes**  x **No**

Certain compressors and filters not supported.

**Implementation name**: Zarr.js

**Date of most recent version**: v0.2.3, 2020-03-27

**Implementation description**: Javascript implementation of Zarr.   
**Implementation URL**: <https://github.com/gzuidhof/zarr.js/>, [http://guido.io/zarr.js/](http://guido.io/zarr.js/#/)

**Is implementation complete**? **Yes**  x **No**

Certain compressors and filters not supported.

*Repeat for each implementation.*

*Optionally, provide a narrative description of the extent of implementation of the proposed Community standard for those proposed standards that are very widely used.*

# Public availability

Is the proposed Community standard currently publicly available? x **Yes** ☐ **No**

URL: <https://zarr.readthedocs.io/en/stable/spec/v2.html>

# Supporting member(s)

*List the supporting organizations. There must be at least three OGC organizations of which at least one must be an OGC Voting Member.*

* UCAR (Ethan Davis)
* NASA (Chris Lynnes)
* USGS (Glenn Guempel)
* UK Met Office (Niall Robinson and Chris Little)
* Columbia University (Bob Chen and Ryan Abernathey)
* ECMWF (Stephan Siemen)

# Intellectual property rights

Will the contributor retain intellectual property rights? x **Yes** ☐ **No**

The Zarr spec copyright is held by the Zarr Developers. However, the spec and all Zarr implementations are licensed under the open-source MIT license, allowing OGC members to use them without restriction.