OGC® DOCUMENT: YY-999

External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

OGC (ADD TITLE TEXT)

STANDARD
Implementation

DRAFT

Version: 1.0
Submission Date: 2029-03-30
Approval Date: 2029-03-30
Publication Date: 2029-03-30
Editor: Christophe Noël, Brianna Pagán

CONTENTS

I ABSTRACT

Zarr provides efficient chunked storage for n-dimensional arrays but do not provide with the semantic constructs required for geospatial and scientific data workflows.

GeoZarr defines an abstract data model and a set of conventions for representing geospatial and scientific datasets in the Zarr format:

- GeoZarr bridges the Unidata CDM and the Zarr format. GeoZarr establishes the link between the Unidata Common Data Model (CDM) and the Zarr format by defining how the semantic constructs of the CDM are represented within Zarr's storage model.
- Supports community metadata standards like CF, GeoTIFF, and GDAL.
- Extends CDM for geospatial through multiscale overviews and affine transformations.

By providing a standardized framework for geospatial semantics, GeoZarr enables scientific and geospatial applications to fully utilize cloud-native storage architectures while maintaining the rich metadata and coordinate referencing required for Earth observation workflows. The result is a modern, scalable approach to storing and accessing geospatial data that meets the needs of both data providers and consumers.

II KEYWORDS

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, API, openapi, html

III PREFACE

The GeoZarr Standard defines a layered, standards-based framework for representing and encoding geospatial and scientific datasets in the Zarr format. The purpose of this document is to provide implementation guidance and normative structure for consistent, interoperable adoption of GeoZarr across tools, platforms, and services. This work extends prior standardisation efforts within the OGC, including OGC API – Tiles, the Tile Matrix Set Standard, and EO metadata conventions, and anticipates integration with catalogue systems such as STAC.

This Standard has been developed in collaboration with contributors from Earth observation, climate science, geospatial analysis, and cloud-native geodata infrastructure communities. Future work may extend this model to additional storage formats, API services, and semantic layers.

IV SECURITY CONSIDERATIONS

No security considerations have been made for this document.

V SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Organization One
- Organization Two

VI SUBMITTERS

All questions regarding this submission should be directed to the editor or the submitters:

Table — Table of submitters

Name	**Affiliation**
Christophe Noël *(editor)*	Spacebel
Brianna Pagán *(editor)*	DevSeed
Ryan Abernathey	EarthMover
TBD	TBD

1

SCOPE

1 SCOPE

The GeoZarr Standard defines a conceptual and implementation framework for representing and encoding geospatial and scientific datasets using the Zarr format. The scope of this Standard includes the definition of a format-agnostic data model, the specification of its encoding into Zarr Version 2 and Version 3, and a set of extensions to support affine transformations and overviews.

These capabilities are necessary for geospatial data because Zarr does not provide semantic constructs for geospatial data interpretation. Applications need to understand not just array shapes and values, but coordinate meanings, projection parameters, and scientific metadata. GeoZarr fills this gap without compromising Zarr's performance characteristics.

1.1. Why GeoZarr Exists

Zarr, by design, is a low-level container for storing n-dimensional arrays and metadata. While this simplicity is a strength for performance and interoperability, it means Zarr lacks higher-level concepts that geospatial applications require:

- **Coordinate Systems:** No native way to associate spatial or temporal meaning with array dimensions
- **Grid Mappings:** No standard mechanism for projection and coordinate reference system metadata
- **Semantic Metadata:** No conventions for units, standard names, or scientific attributes
- **Variable Relationships:** No formal distinction between coordinate variables and data variables

These concepts are essential for geospatial workflows but must be layered on top of Zarr's array storage. GeoZarr provides this semantic layer through proven standards (Common Data Model and CF conventions) while preserving Zarr's cloud-native advantages.

1.2. Relationship to Zarr Core Concepts

GeoZarr builds upon Zarr's foundational concepts of stores and hierarchies. A Zarr store provides the storage and retrieval interface (e.g., filesystem, cloud object storage), while a hierarchy defines the logical tree structure of groups and arrays within that store. GeoZarr specifies how to organize and structure hierarchies to support geospatial semantics, without modifying the underlying store interface.

1.3. Use Cases and Applications

This Standard addresses the needs of Earth observation, environmental monitoring, and geospatial analysis applications that require efficient, scalable access to multidimensional datasets. It enables the harmonisation of existing data models with operational encoding formats suitable for cloud-native storage and analysis.

Typical use cases include: * Storage and processing of raster and gridded data * Management of data cubes with temporal or vertical dimensions * Integration with catalogue systems through standardized metadata * Multi-resolution tiling for efficient visualization and analysis * Cloud-optimized access to large geospatial datasets

2

CONFORMANCE

2 CONFORMANCE

WARNING

This section should be ignored and requirements classes should be designed and summarized here once the specification is completed.

The GeoZarr Unified Data Model is structured around a modular set of requirements classes. These classes define the conformance criteria for datasets and implementations adopting the GeoZarr specification. Each class provides a distinct set of structural or semantic expectations, facilitating interoperability across a broad spectrum of geospatial and scientific use cases.

The **Core** requirements class defines the minimal compliance necessary to claim conformance with the GeoZarr Unified Data Model. It is intentionally open and permissive, supporting incremental adoption and broad compatibility with existing Zarr tools and data models based on the Unidata Common Data Model (CDM).

Additional requirements classes are defined to support enhanced functionality, semantic richness, and interoperability with established geospatial conventions and systems. These include extensions for time series, coordinate systems, affine transformations, and multiscale tiling.

Table 1 — Requirements Classes Overview

REQUIREMENTS CLASS	DESCRIPTION	IDENTIFIER
Core Model	Specifies minimum conformance for encoding multidimensional datasets in Zarr using CDM-aligned constructs. Includes dimensions, variables, attributes, and groups.	`http://www.opengis.net/spec/geozarr/1.0/conf/core`
Time Series Support	Defines conventions for temporal dimensions and time coordinate variables to support time-aware arrays.	`http://www.opengis.net/spec/geozarr/1.0/conf/time`
Coordinate Reference Systems	Specifies use of CF-compliant CRS metadata, including `grid_mapping`, `standard_name`, and EPSG codes.	`http://www.opengis.net/spec/geozarr/1.0/conf/crs`
GeoTransform Metadata	Enables affine spatial referencing via GDAL-compatible `GeoTransform` metadata and optional interpolation hints.	`http://www.opengis.net/spec/geozarr/1.0/conf/geotransform`
Multiscale Overviews	Specifies multiscale tiled layout using zoom levels and Tile Matrix Sets as per OGC API – Tiles.	`http://www.opengis.net/spec/geozarr/1.0/conf/overviews`

REQUIREMENTS CLASS	DESCRIPTION	IDENTIFIER
STAC Metadata Integration	Allows embedding or referencing of STAC Collection/Item metadata for discovery and indexing.	`http://www.opengis.net/spec/geozarr/1.0/conf/stac`
Projection Coordinates	Supports encoding of data in projected coordinate systems and association with spatial reference metadata.	`http://www.opengis.net/spec/geozarr/1.0/conf/projected`
Spectral Bands	Defines conventions for encoding multi-band imagery, including band identifiers, wavelengths, and metadata attributes.	`http://www.opengis.net/spec/geozarr/1.0/conf/bands`

Each requirements class is independently defined. Implementations may declare conformance with any subset of classes appropriate to their use case. All classes build upon the Core model.

Associated conformance tests for each class are detailed in Annex A.

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Miles, A., et al.: *Zarr Specification Version 2*. Zarr Developers. https://zarr.readthedocs.io/en/stable/spec/v2.html

Zarr Community: *Zarr Specification Version 3*. https://zarr-specs.readthedocs.io/en/latest/v3.0

Unidata: *The Common Data Model*. https://docs.unidata.ucar.edu/netcdf-java/5.0/userguide/common_data_model_overview.html

Rew, R., Davis, G.: *NetCDF: An Interface for Scientific Data Access*. IEEE Computer Graphics and Applications, 10(4), 76–82 (1990). https://doi.org/10.1109/38.56302

CF Community: *Climate and Forecast (CF) Metadata Conventions, Version 1.10*. https://cfconventions.org/

GDAL Developers: *GDAL/OGR Version 3.8 Documentation*. Open Source Geospatial Foundation. https://gdal.org

Open Geospatial Consortium: *OGC Two Dimensional Tile Matrix Set and Tile Pyramid* (OGC 17-083r2). https://docs.ogc.org/is/17-083r2/17-083r2.html

STAC Community: *STAC Specification v1.0.0*. https://stacspec.org/en/

Open Geospatial Consortium: *OGC Compliance Testing Policies and Procedures*, OGC 08-134r10. https://portal.ogc.org/files/?artifact_id=55184

4

TERMS, DEFINITIONS AND ABBREVIATED TERMS

4 TERMS, DEFINITIONS AND ABBREVIATED TERMS

This document uses the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

4.1. Terms and definitions

GeoZarr specification inherits the terms from the following sources:

- Unidata Common Data Model
- Zarr concepts and terminology.

4.1.1. **affine transformation**

An affine transformation is a geometric mapping that preserves points, straight lines, and parallelism. It combines linear transformations (such as rotation, scaling, reflection, or shear) with translation.

4.1.2. **array**

A multidimensional, regularly spaced collection of values (e.g., raster data or gridded measurements), typically indexed by dimensions such as time, latitude, longitude, or spectral band.

4.1.3. chunk

A sub-array representing a partition of a larger array, used to optimize data access and storage. In Zarr, data is stored and accessed as a collection of independently compressed chunks.

4.1.4. coordinate variable

A one-dimensional array whose values define the coordinate system for a dimension of one or more data variables. Typical examples include latitude, longitude, time, or vertical levels.

4.1.5. data model

A data model is an **abstract**, conceptual framework that defines how data is structured, organized, and interpreted, independent of any particular storage medium or implementation. In contrast, a file format represents a concrete realization of this model, defining how the data is stored on disk.

4.1.6. data variable

An array containing the primary geospatial or scientific measurements of interest (e.g., temperature, reflectance). Data variables are defined over one or more dimensions and associated with attributes.

4.1.7. dimension

An index axis along which arrays are organised. Dimensions provide a naming and ordering scheme for accessing data in multidimensional arrays (e.g., `time`, `x`, `y`, `band`).

4.1.8. dataset

Avoid using: this term is overloaded and avoided in this document. A dataset usually represent a self-contained group of variables within a hierarchical data structure. They often share one or more dimensions and represent the unit that can be opened by a data access library (see variable group)

4.1.9. metadata

Structured information describing the content, context, and semantics of datasets, variables, and attributes. GeoZarr metadata includes CF attributes, geotransform definitions, and links to STAC metadata where applicable.

4.1.10. overview

A downscaled representation of a variable that facilitates rapid data display and efficient zooming. Overviews provide lower-resolution versions of the original data, enabling quick visualization and access without reading the full-resolution array. Multiple overview levels may be generated to support progressive rendering across different scales.

4.1.11. store

A system that provides storage and retrieval operations for Zarr hierarchies, as defined in the Zarr core specification. A store implements the abstract store interface and can be backed by various storage technologies such as filesystems, cloud object storage, or databases. GeoZarr hierarchies are stored within and accessed through Zarr stores.

4.1.12. tile matrix set

A spatial tiling scheme defined by a hierarchy of zoom levels and consistent grid parameters (e.g., scale, CRS). Tile Matrix Sets enable spatial indexing and tiling of gridded data.

4.1.13. **variable group**

A variable group is a container that includes a coherent collection of variables sharing the same dimensional structure and coordinate system (and may contain additional variables or subgroups). It is conceptually equivalent to an xarray Dataset..

4.2. Abbreviated terms

API	Application Programming Interface
CDM	Common Data Model
CF	Climate and Forecast Conventions
CRS	Coordinate Reference System
EPSG	European Petroleum Survey Group
GDAL	Geospatial Data Abstraction Library
GeoTIFF	Georeferenced Tagged Image File Format
JSON	JavaScript Object Notation
OGC	Open Geospatial Consortium
STAC	SpatioTemporal Asset Catalog
UDM	Unified Data Model
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
Zarr	Zipped Array Storage format

5

CONVENTIONS

5 CONVENTIONS

This section describes the conventions used throughout this Standard, including identifiers, metadata schemas, and referencing mechanisms relevant to the GeoZarr Unified Data Model.

5.1. Identifiers

The normative provisions in this Standard are denoted by the base URI:

`http://www.opengis.net/spec/geozarr/1.0`

All requirements, recommendations, permissions, and conformance tests that appear in this document are assigned relative URIs anchored to this base.

For example:

`http://www.opengis.net/spec/geozarr/1.0/conf/core` — refers to the Core Requirements Class of the GeoZarr Unified Data Model.

5.2. Data Encoding

This Standard specifies the encoding of geospatial data in the Zarr format. Zarr is a chunked, compressed, binary format for n-dimensional arrays, with support for both Version 2 and Version 3 encodings.

The specification makes extensive use of:

- `zarr.json` metadata documents (Zarr v3)
- `.zgroup`, `.zattrs`, `.zarray` metadata files (Zarr v2)
- JSON-compatible structures for metadata, attributes, and conformance declarations

5.3. Schemas

Metadata schemas referenced in this Standard are represented using JSON-compatible objects and may be defined formally using JSON Schema. Metadata structures for tile matrix sets, STAC properties, or CF metadata may be embedded inline or referenced externally via URI.

5.4. URI Usage

URIs used in this Standard must comply with [RFC3986] (URI Syntax). When including reserved characters in a URI, they must be percent-encoded. Dataset identifiers, metadata links, and STAC references should use persistent and canonical forms to support reproducibility and catalogue integration.

6

OVERVIEW

6 OVERVIEW

The **GeoZarr Standard** defines an **abstract data model** and a set of **conventions** for representing and describing geospatial and scientific datasets using the **Zarr** format.

Zarr provides efficient, chunked storage for n-dimensional arrays but does not include the semantic constructs required for geospatial and scientific data workflows. The **Unidata Common Data Model (CDM)** addresses this gap by introducing essential concepts that structure information through **variables**, **groups**, **coordinates**, and **metadata**. This abstract data model provides the semantic framework that enables structured interpretation of array-based data on top of Zarr's storage foundation.

The **primary objective** of GeoZarr is to specify how the **CDM** is encoded within Zarr. GeoZarr provides normative rules for encoding these CDM concepts in Zarr and thereby standardises the encoding practices already adopted by CDM-compatible libraries such as **xarray** and **nczarr**, promoting consistent interpretation and interoperability across tools and platforms.

By defining an **abstract model** based on the **CDM** and a corresponding **encoding for Zarr**, GeoZarr establishes an explicit relationship between **the conceptual structure of the data** and **its physical storage representation**. Zarr defines how data are stored and accessed as chunked, hierarchical arrays, while GeoZarr specifies how this stored structure represents the scientific and geospatial meaning of the dataset..

As a **secondary objective**, GeoZarr extends the **CDM base layer** with additional capabilities required for geospatial and cloud-native applications. These extensions include **multiscale overviews**, which enable the representation of data at multiple levels of detail, and **affine transformations**, which define the spatial relationship between data coordinates and real-world locations. All extensions remain fully aligned with the CDM framework.

The **CDM** base layer also provides a **generic framework** capable of hosting metadata from a wide range of community standards. GeoZarr encourages the use of the **Climate and Forecast (CF) Conventions**, which are themselves defined around the CDM model, without imposing them as mandatory. This flexibility also supports metadata from other domain-specific standards such as **GeoTIFF**, **GDAL**, and similar geospatial conventions.

7

GEOZARR DATA MODEL

7 GEOZARR DATA MODEL

7.1. Scope and Purpose

The GeoZarr Data Model defines the abstract structure for representing geospatial and scientific gridded data within the GeoZarr framework.

The GeoZarr Data Model serves the following purposes:

- to **clarify the role of the Common Data Model (CDM)** as the structural foundation (recognising its suitability for Zarr as demonstrated by **xarray**, **GDAL**, and **nczarr**);
- to **extend the CDM** with additional geospatial capabilities required for cloud-native and tiled data representations, including **affine transformations** and **multiscale overviews**;
- to **ensure compatibility** with established data models and conventions such as **netCDF**, **CF**, **GDAL**, and **GeoTIFF**.

7.2. Conceptual Basis

The GeoZarr Data Model adopts and extends the **Unidata Common Data Model (CDM)** as its conceptual foundation. The CDM defines a hierarchy of **groups**, **variables**, **dimensions**, and **attributes** that together describe the logical organisation of scientific data.

GeoZarr reuses these constructs and introduces an additional layer of **GeoZarr Extensions** that provide explicit geospatial semantics and support for cloud-native scalability: **Affine transformations defining spatial reference through linear mapping between array indices and real-world coordinates;** **Overviews** enabling multiscale representations and efficient visualization of large datasets.

7.3. Format Independence

Although the model was defined to support encoding in **Zarr**, it remains **format-agnostic** at the conceptual level. Implementations may serialise the same GeoZarr Data Model structure into other compatible encodings, such as NetCDF or alternative object-based formats, provided they preserve the semantics and conformance requirements defined herein.

This separation between **conceptual model** and **physical encoding** ensures that GeoZarr can evolve alongside emerging storage technologies while maintaining interoperability with existing CDM- and CF-based infrastructures.

7.4. Conceptual Layers

The GeoZarr Data Model is organised into two conceptual layers:

1. the **Common Data Model (CDM)** – structural foundation for multidimensional data;
2. the **GeoZarr Extensions** – additional constructs for geospatial semantics and multiscale representations.

7.4.1. Common Data Model (CDM)

The **Unidata Common Data Model (CDM)** defines the logical structure of scientific datasets through a hierarchy of **Groups**, **Variables**, **Dimensions**, and **Attributes**. It provides the foundation upon which GeoZarr and many existing libraries (such as **xarray**, **GDAL**, and **nczarr**) operate.

```
@startuml
class Group {
}

class Variable {
}

class Dimension {
}

class Attribute {
}

class DataArray {
}

class CoordinateVariable {
}

class Subgroup {
}

Group "1" *-- "0..*" Variable
Group "1" *-- "0..*" Dimension
Group "1" *-- "0..*" Attribute
Group "1" *-- "0..*" Subgroup
Variable "1" *-- "1" DataArray
Variable <|-- CoordinateVariable
```

```
@enduml
```

Listing 1

- A **Group** is a container that may include variables, dimensions, attributes, and subgroups.
- A **Variable** represents a multidimensional array associated with one or more dimensions and attributes.
- A **Dimension** defines an index axis used to organise data within variables.
- An **Attribute** holds descriptive metadata for groups or variables.
- A **Coordinate Variables** supplies coordinate values along dimensions, establishing spatial or temporal context.
- A **Data Array** represents observed or simulated phenomena, associated with dimensions and coordinate variables.

This structure enables consistent representation of scientific data independently of storage format, providing the base semantic framework for all GeoZarr encodings.

7.4.2. GeoZarr Extensions

GeoZarr extends the CDM with additional geospatial constructs required for cloud-native applications:

- **Affine transformations** — define the mapping between array indices and real-world coordinates using linear coefficients. This enables compact georeferencing for regularly gridded data.
- **Multiscale overviews** — represent downsampled versions of variables for efficient visualisation and scalable access. Overviews are structured as subordinate variable groups sharing the same coordinate system.

All extensions remain aligned with the CDM hierarchy and are encoded using the same core constructs (groups, variables, and attributes). Together, they provide the minimal geospatial extensions necessary for efficient, standards-based representation of Earth observation and scientific data in cloud environments.

```
@startuml
skinparam classAttributeIconSize 0
skinparam linetype ortho
skinparam packageStyle rectangle
skinparam backgroundColor #FFFFFF
title GeoZarr Extensions — Geospatial Enhancements

package "Common Data Model (CDM)" {
  class Group
  class Variable
  class Dimension
  class Attribute
}
```

```
package "GeoZarr Extensions" {
  class AffineTransform
  class Overview
}

Variable --> AffineTransform : georeference
Group --> Overview : provides
Variable --> Overview : provides

@enduml
```

Listing 2

7.5. Interoperability with Other Frameworks

The Common Data Model (CDM), with its flexible hierarchy of groups, variables, dimensions, and attributes allows direct representation of metadata constructs used across multiple scientific and geospatial standards.

This design enables the CDM—and therefore GeoZarr—to act as a **host model** for conventions and metadata originating from other frameworks, while preserving their semantics within a unified structure.

- **netCDF and the Enhanced Data Model** – The netCDF Enhanced Data Model and the CDM share common origins and are conceptually aligned. Both organise data into variables, dimensions, and attributes. As a result, most netCDF datasets can be represented as CDM hierarchies without loss of structure or metadata. Conversely, GeoZarr datasets that follow the CDM pattern can be serialised as valid netCDF encodings.

- **CF Conventions** – The CF data model is encoding independent but conceptually compatible with the CDM. CF metadata constructs—such as coordinate and auxiliary coordinate variables, standard names, units, and grid mappings—map directly onto CDM variables and attributes. This allows GeoZarr datasets to incorporate CF semantics naturally, achieving partial or full CF compliance without modifying the underlying data model.

- **GDAL Metadata and Geotransform** – GDAL expresses georeferencing through affine transformation coefficients and projection information. These map directly to GeoZarr extension attributes (for affine transforms and CRS) stored as CDM attributes. GDAL domain metadata can likewise be represented as CDM attributes within groups or variables, maintaining equivalence between GDAL and GeoZarr geospatial metadata.

- **GeoTIFF Tags and Metadata** – GeoTIFF georeferencing information, including coordinate reference system definitions, tie points, and pixel scale, correspond closely to the affine transform and CRS constructs in the GeoZarr Extensions. These elements can be represented as attributes within CDM-compliant groups and variables, ensuring semantic consistency between file-based and cloud-native representations.

Through these mappings, the CDM acts as a **common semantic framework** that integrates metadata from diverse geospatial standards. This interoperability ensures that GeoZarr can serve as both a native storage model and a bridge between existing ecosystems such as netCDF/CF, GDAL, and GeoTIFF.

NOTE: GeoZarr does **not** define the mappings to CDM for metadata from existing conventions or formats such as CF, netCDF, GDAL, or GeoTIFF. These mappings are already established and maintained by widely used libraries and implementations, including **xarray**, **netCDF-Java**, **GDAL**, etc. The role of GeoZarr is to provide a **data model and encoding framework**, not to redefine or replicate existing translation logic between metadata standards.

Accordingly, the **GeoZarr encoding specification** will only prescribe additional rules where a specific encoding behaviour in **Zarr** is required for interoperability or conformance.

7.5.1. Metadata and Discovery Integration

STAC compatibility enables integration with catalogue services for discovery and indexing. Datasets can expose STAC-compliant metadata alongside core metadata, supporting federated search and filtering via STAC APIs.

This approach enables seamless integration into modern data catalogues and platforms that support EO discovery standards.

7.5.2. Tool and Ecosystem Support

The Unified Data Model facilitates interoperability with tools and libraries across the following domains:

- **Scientific computing**: NetCDF-based libraries (e.g., xarray, netCDF4), Zarr-compatible clients.
- **Geospatial processing**: GDAL, rasterio, QGIS (via Zarr driver extensions or translations).
- **Cloud-native infrastructure**: support for parallel access, chunked storage, and hierarchical grouping compatible with object storage.

Tooling support is expected to grow via standard-conformant implementations, easing adoption across domains and infrastructures.

8

ENCODINGS FOR ZARR

8 ENCODINGS FOR ZARR

This clause defines the normative mapping between the **GeoZarr Data Model** and the **Zarr storage format**. It specifies how the structural elements of the **Common Data Model (CDM)** — groups, variables, dimensions, and attributes — are encoded in **Zarr v2** and **Zarr v3**, and identifies additional constraints introduced by GeoZarr.

GeoZarr's encoding rules are limited to cases where explicit guidance is required for interoperability. GeoZarr does **not** redefine how CF, GDAL, or other metadata conventions map to CDM constructs — these mappings are already implemented in community libraries such as **xarray**, **GDAL**, and **netCDF-Java**.

The GeoZarr encoding rules therefore focus on **CDM structure and semantics**, with additional subsections specifying any **Zarr-specific requirements** for supported metadata conventions.

8.1. Common Data Model Encodings

8.1.1. Hierarchical Structure

A GeoZarr hierarchy follows the CDM model of a tree of **groups**, **variables** (arrays), **dimensions**, and **attributes**. Each Zarr store contains a single root group and an arbitrary number of child groups and arrays, organised recursively.

Table 2

CDM ELEMENT	ZARR V2 ENCODING	ZARR V3 ENCODING
Group	Directory with `.zgroup` and `.zattrs`	Directory containing `zarr.json` with `"node_type": "group"`
Variable (Array)	Directory with `.zarray` and `.zattrs`	Directory containing `zarr.json` with `"node_type": "array"`
Attributes	`.zattrs` file (JSON object)	`attributes` field in `zarr.json`

Zarr v3 nodes must declare `"zarr_format": 3` and include `"node_type"` set to either `"group"` or `"array"`. All user-defined metadata, including GeoZarr attributes, shall be placed within the `attributes` field.

8.1.2. Dimensions

Dimensions define the index axes for variables.

Table 3

ASPECT	ZARR V2	ZARR V3
Declaration	`_ARRAY_DIMENSIONS` attribute in `.zattrs`	`dimension_names` field in `zarr.json`
Scope	Implicit, per array	Explicit, per array; names are globally unique within a group hierarchy

Example (Zarr v3 array with two dimensions):

```
{
  "zarr_format": 3,
  "node_type": "array",
  "shape": [180, 360],
  "dimension_names": ["lat", "lon"],
  ...
}
```

Listing 3

Shared Dimensions:

Zarr does not define dimension entities as standalone objects. To preserve CDM semantics, GeoZarr requires that dimension names be **unique within each group hierarchy** and reused consistently across variables that share the same axis.

As in **netCDF-4**, where groups can define their own local dimensions, GeoZarr allows dimensions to be scoped within groups. When a dimension defined in one group is shared by variables located in descendant groups, implementations may indicate this relationship by prefixing the dimension name with a slash (e.g., `"/time"`). In this context, the leading slash signifies that the dimension is defined in an **ancestor group**—not necessarily the root of the hierarchy—and should be interpreted as a shared axis accessible to all subordinate groups.

8.1.3. Coordinate Variables

Coordinate variables define the spatial, temporal, or other contextual axes for data variables. They are stored as one-dimensional arrays associated with their corresponding dimensions.

Table 4

ASPECT	ZARR V2	ZARR V3
Storage	Zarr array with `.zarray`, `.zattrs`	Zarr array with `zarr.json`
Dimension Binding	`_ARRAY_DIMENSIONS` in `.zattrs`	`dimension_names` in `zarr.json`
Metadata	CF-style attributes (e.g., `standard_name`, `units`, `axis`)	Same under `attributes`

Example (Zarr v3 coordinate array):

```
{
  "zarr_format": 3,
  "node_type": "array",
  "shape": [180],
  "dimension_names": ["lat"],
  "data_type": "float32",
  "attributes": {
    "standard_name": "latitude",
    "units": "degrees_north",
    "axis": "Y"
  }
}
```

Listing 4

Coordinate variables may also reference **grid mapping** variables for coordinate reference systems, as defined in the CF conventions.

8.1.4. Data Variables

Data variables represent primary measurements or derived quantities. They are encoded as multidimensional arrays linked to one or more dimensions and accompanied by descriptive metadata.

Table 5

ASPECT	ZARR V2	ZARR V3
Storage	Directory containing `.zarray` and `.zattrs`	Directory containing `zarr.json` with `"node_type": "array"`
Dimension Binding	`_ARRAY_DIMENSIONS` attribute	`dimension_names` field
Metadata	Attributes such as `standard_name`, `units`, `long_name`, `_FillValue`, `scale_factor`, `add_offset`	Same, with typed attributes permitted in v3

Example:

```
{
  "zarr_format": 3,
  "node_type": "array",
  "shape": [12, 180, 360],
  "dimension_names": ["time", "lat", "lon"],
  "attributes": {
    "standard_name": "air_temperature",
    "units": "K",
    "long_name": "Surface air temperature",
    "_FillValue": -9999.0
  }
}
```

Listing 5

8.1.5. Global and Group Metadata

Metadata applying to the entire hierarchy or subgroup is stored at the group level.

Table 6

ASPECT	ZARR V2	ZARR V3
Location	`.zattrs` in root group	`attributes` field in root `zarr.json`
Identification	`.zgroup` file	`"node_type": "group"`
Conventions	`Conventions` attribute (e.g., `CF-1.10`)	Same under `attributes`

Example:

```
{
  "zarr_format": 3,
  "node_type": "group",
  "attributes": {
    "title": "Example Dataset",
    "summary": "Multidimensional Earth Observation data",
    "institution": "Example Space Agency",
    "Conventions": "CF-1.10"
  }
}
```

Listing 6

8.1.6. Variable and Attribute Metadata

All metadata attributes for groups, coordinate variables, and data variables should follow established community naming and typing conventions. GeoZarr encourages CF-compliant naming where applicable but does not require it.

Attributes shall: - use UTF-8–encoded names; - have JSON-compatible values (string, number, boolean, or array); - remain consistent across group hierarchies.

Typical attributes include:

- CF: `standard_name`, `units`, `axis`, `grid_mapping`
- Generic: `_FillValue`, `scale_factor`, `add_offset`, `long_name`, `missing_value`
- GDAL-compatible: `spatial_ref`, `GeoTransform`, `AREA_OR_POINT`

Structured metadata values, such as JSON or XML content, may be included directly as objects rather than as serialised text. Implementations are encouraged to **store such metadata in deserialised form** (as native JSON objects) whenever possible, ensuring that attributes remain machine-readable and conform to JSON type rules.

If serialised representations (e.g., XML strings or JSON text blocks) are used, they shall be valid UTF-8 strings and clearly identified by attribute naming or context.

8.1.7. CDM Encoding Notes and Special Cases

- **Shared Dimensions** – To emulate the CDM concept of shared dimensions, GeoZarr requires that identical dimension names across arrays refer to the same logical axis. Libraries implementing GeoZarr should preserve this relationship explicitly in their in-memory representations.
- **Unlimited Dimensions** – Zarr's chunked structure inherently supports extensible dimensions. A dimension can be declared unlimited by allowing its corresponding array dimension to grow dynamically (e.g., time). The use of `"resizeable": true` (Zarr v3) or dynamic chunk append operations is recommended.
- **Nested Groups and Subgroups** – Zarr v3 groups may nest recursively. Each subgroup represents a CDM group and may hold its own variables and attributes. This structure supports logical organisation such as multiple collections, products, or resolution levels.

8.1.8. Metadata Integration for CF, GDAL, and GeoTIFF

While the GeoZarr Data Model provides the structure for metadata storage, **GeoZarr does not redefine how CF, GDAL, or GeoTIFF metadata are mapped into this structure**. These mappings are well established in community libraries (e.g., **xarray**, **netCDF-Java**, **GDAL**).

GeoZarr encoding rules therefore only specify **when a specific Zarr encoding requirement applies**, such as:

- use of `attributes` fields in `zarr.json` for CF or GDAL metadata;
- preservation of key metadata names (`grid_mapping`, `spatial_ref`, `GeoTransform`);

- ensuring metadata values remain valid JSON types.

Implementations may rely on existing libraries to populate or interpret such metadata consistently.

8.2. Encoding of Multiscale Overviews in Zarr

This clause specifies how multiscale tiling (also known as overviews or pyramids) is encoded in Zarr hierarchies conforming to the Unified Data Model. The encoding supports both Zarr Version 2 and Version 3 and is aligned with the OGC Two Dimensional Tile Matrix Set Standard.

A multiscale group contains one or more child groups, where each child group is a dataset representing a zoom level of the data. Additional resolution levels can be added over time, with each new level storing a coarser-resolution resampled version of the original data variables.

8.2.1. Hierarchical Layout

Each zoom level SHALL be represented as a child group, identified by the Tile Matrix identifier (e.g., `"0"`, `"1"`, `"2"`). These child groups SHALL be organized hierarchically under a common multiscale group and each SHALL be a dataset containing the complete set of variables (arrays) corresponding to that resolution. All zoom-level datasets MUST maintain consistent structure.

Table 7

STRUCTURE	ZARR V2	ZARR V3
Zoom level datasets	Subdirectories with `.zgroup` and `.zattrs`	Subdirectories with `zarr.json`, `node_type: group`
Variables at each level	Arrays (`.zarray`, `.zattrs`) in each dataset	Arrays (`zarr.json`, `node_type: array`) in each dataset
Multiscale metadata	`multiscales` defined in multiscale group `.zattrs`	`multiscales` defined in multiscale group `zarr.json` under `attributes`

Each zoom-level dataset MUST define chunking (tiling) along the spatial dimensions (`X`, `Y`, or `lon`, `lat`). Recommended chunk sizes are 256×256 or 512×512.

8.2.2. Metadata Encoding

Multiscale metadata SHALL be defined using a `multiscales` attribute located in the multiscale group. This attribute SHALL be a JSON object with the following members:

- `tile_matrix_set` – Identifier, URI, or inline JSON object compliant with OGC TileMatrixSet v2
- `resampling_method` – One of the standard string values (e.g., `"nearest"`, `"average"`)
- `tile_matrix_set_limits` – (optional) Zoom-level limits following the STAC Tiled Asset style

8.2.2.1. Zarr v2 Encoding Example (`.zattrs`)

```
{
  "multiscales": {
    "tile_matrix_set": "WebMercatorQuad",
    "resampling_method": "nearest"
  }
}
```

Listing 7

8.2.2.2. Zarr v3 Encoding Example (`zarr.json`)

```
{
  "zarr_format": 3,
  "node_type": "group",
  "attributes": {
    "multiscales": {
      "tile_matrix_set": "WebMercatorQuad",
      "resampling_method": "nearest"
    }
  }
}
```

Listing 8

8.2.3. Tile Matrix Set Representation

The `tile_matrix_set` member MAY take one of the following forms:

- A string referring to a well-known identifier (e.g., `"WebMercatorQuad"`)
- A URI pointing to a JSON document describing the tile matrix set
- An inline JSON object (CamelCase, OGC TMS 2.0 compatible)

Zoom level identifiers in the tile matrix set MUST match the names of the child groups. The spatial reference system declared in `supportedCRS` MUST match the one declared in the corresponding `grid_mapping` of the data variables.

8.2.4. Chunk Layout Alignment

At each zoom level, chunking SHALL match the tile layout defined by the TileMatrix:

- Chunks MUST be aligned with the tile grid (1:1 mapping between chunks and tiles)
- Chunk sizes MUST match the `tileWidth` and `tileHeight` declared in the TileMatrix
- Spatial dimensions MUST be clearly identified using `dimension_names` (v3) or `_ARRAY_DIMENSIONS` (v2)

8.2.5. Tile Matrix Set Limits

The `tile_matrix_set_limits` object MAY define the extent of actual data coverage for each zoom level. This follows the style of the STAC tiled-assets extension rather than the full OGC JSON encoding.

Example:

```
"tile_matrix_set_limits": {
  "1": {
    "min_tile_col": 0,
    "max_tile_col": 1,
    "min_tile_row": 0,
    "max_tile_row": 1
  }
}
```

Listing 9

8.2.6. Resampling Method

The `resampling_method` MUST indicate the method used for downsampling across zoom levels. The value MUST be one of:

`nearest`, `average`, `bilinear`, `cubic`, `cubic_spline`, `lanczos`, `mode`, `max`, `min`, `med`, `sum`, `q1`, `q3`, `rms`, `gauss`

The same method MUST apply across all levels.