OGC® DOCUMENT: YY-999

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

Open Geospatial Consortium

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

The GeoZarr Unified Data Model and Encoding Standard specifies a conceptual and implementation framework for representing multidimensional, geospatial datasets using the Zarr format. This Standard builds upon the Unidata Common Data Model (CDM) and the Climate and Forecast (CF) Conventions, and introduces interoperable constructs for tiling, georeferencing, and metadata integration.

The model defines core elements—dimensions, coordinate variables, data variables, attributes—and optional extensions for multi-resolution overviews, affine geotransforms, and STAC metadata. Encoding guidance is provided for Zarr Version 2 and Zarr Version 3, including chunking, group hierarchy, and metadata conventions.

GeoZarr aims to bridge scientific and geospatial communities by enabling round-trip transformations with formats such as NetCDF and GeoTIFF, and supporting compatibility with tools in the scientific Python and geospatial ecosystems. This Standard enables scalable, standards-compliant, and semantically rich data structures for cloud-native Earth observation applications.

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 Unified Data Model and Encoding Standard defines a layered, standards-based framework for representing and encoding geospatial and scientific datasets in the Zarr format. It integrates foundational specifications such as the Unidata Common Data Model (CDM), the CF Conventions, and selected OGC and community standards to enable semantic, structural, and operational interoperability across Earth observation platforms and geospatial ecosystems.

This Standard introduces a unified model that harmonises metadata structures, array-based data representations, coordinate referencing, and multiscale tiling semantics. It provides a coherent framework that facilitates encoding into Zarr v2 and v3, supporting scalable, cloud-native workflows.

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 Unified Data Model and Encoding 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 unified data model, the specification of its encoding into Zarr Version 2 and Version 3, and the establishment of extension points to support interoperability with external metadata and tiling standards.

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, such as the Unidata Common Data Model (CDM) and the Climate and Forecast (CF) Conventions, with operational encoding formats suitable for cloud-native storage and analysis.

Typical use cases include the storage, transformation, discovery, and processing of raster and gridded data, data cubes with temporal or vertical dimensions, and catalogue-enabled datasets integrated with metadata standards such as STAC and OGC Tile Matrix Sets.

2

CONFORMANCE

2 CONFORMANCE

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`
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`

REQUIREMENTS CLASS	DESCRIPTION	IDENTIFIER
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

4.1.1. 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.

4.1.2. chunk

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

4.1.3. 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.

4.1.4. **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.

4.1.5. **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`).

4.1.6. **group**

A container for datasets, variables, dimensions, and metadata in Zarr. Groups may be nested to represent a logical hierarchy (e.g., for resolutions or collections).

4.1.7. **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.

4.1.8. **multiscale dataset**

A dataset that includes multiple representations of the same data variable at varying spatial resolutions. Each resolution level is associated with a tile matrix from an OGC Tile Matrix Set.

4.1.9. 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.

4.1.10. transform

An affine transformation used to convert between grid coordinates and geospatial coordinates, typically defined using the GDAL GeoTransform convention.

4.1.11. unified data model (UDM)

A conceptual model that defines how to structure geospatial data in Zarr using CDM-based constructs, including support for coordinate referencing, metadata integration, and multiscale representations.

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 Unified Data Model and Encoding Standard defines a conceptual and implementation framework for representing multidimensional geospatial data using the Zarr format. Developed under the guidance of the OGC GeoZarr Standards Working Group (SWG), the Standard establishes conventions for encoding scientific and Earth observation datasets in a way that promotes scalability, interoperability, and compatibility with cloud-native infrastructure.

GeoZarr is built on widely adopted community standards, including the Unidata Common Data Model (CDM) and Climate and Forecast (CF) Conventions. It introduces additional extensions and structural constructs to support multi-resolution tiling, geospatial referencing, and catalogue-enabled metadata integration (e.g., STAC).

This Standard provides both:

- **Core requirements**, which define minimal compliance to represent array-based datasets using CDM constructs in Zarr, supporting open and permissive adoption across use cases.
- **Modular extension classes**, which define additional capabilities such as time series support, affine geotransform referencing, multi-resolution overviews, and projection coordinates, in line with OGC and community practices.

These modular components enable GeoZarr to serve a wide range of applications—from basic EO data storage to high-performance, cloud-native visualisation and analytics workflows.

6.1. Encodings

GeoZarr supports encoding in both Zarr Version 2 and Zarr Version 3. Each version defines how arrays, groups, and metadata are stored within a directory-based structure. All metadata is encoded in JSON-compatible formats, ensuring both human readability and machine interoperability.

Encoding guidelines include:

- Hierarchical grouping of datasets via Zarr groups.
- Dimension indexing and binding via dimension metadata.
- Attribute-based metadata compliant with CF conventions.
- Multi-resolution overviews aligned with OGC Tile Matrix Sets.
- Optional integration of STAC metadata for discovery and cataloguing.

JSON is the primary format for metadata, attributes, and structural declarations. Implementations are encouraged to support standardised naming conventions, EPSG code references, and structured metadata to facilitate search, validation, and transformation across platforms.

GeoZarr does not prescribe a single interface for data access. Instead, it enables **serverless and cloud-native** data access strategies by aligning its model with chunked, parallelisable storage patterns that are optimised for use in object stores and analytical environments.

7

UNIFIED DATA MODEL

7 UNIFIED DATA MODEL

7.1. Scope and Purpose

This Standard defines a unified data model (UDM) that provides a conceptual framework for representing geospatial and scientific data in Zarr. The purpose of this model is to support standards-based interoperability across Earth observation systems and analytical environments, while preserving compatibility with existing data models and software ecosystems..

The unified data model incorporates and extends the following established specifications and community standards:

- **Unidata Common Data Model (CDM)** – Provides the foundational resource structure for scientific datasets, encompassing dimensions, coordinate systems, variables, and associated metadata elements.
- **CF (Climate and Forecast) Conventions** – Defines a widely adopted metadata profile for describing spatiotemporal semantics in CDM-based datasets.
- **Selected constructs from related Standards and practices**, including:
- The **OGC Tile Matrix Set Standard**, which enables multi-resolution representations of gridded data.
- **GDAL geotransform metadata**, used to express affine transformations and interpolation characteristics.
- **SpatioTemporal Asset Catalog (STAC)** metadata elements for resource discovery and cataloguing (Collection and Item constructs).

The unified model is format-agnostic and describes the abstract structure of resources independently of the physical encoding. It does not redefine the semantics of the CDM or CF conventions, but introduces integration and extension points required to support tiled multiscale data, geospatial referencing, and metadata for discovery.

This clause specifies the logical composition of the unified model, the external standards it leverages, and the conformance points that facilitate harmonised implementation within the GeoZarr framework.

7.2. Foundational Model and Standards Reuse

The unified data model described in this Standard is derived from established community specifications to maximise interoperability and to enable the reuse of mature tools and practices. The model is grounded in the Unidata Common Data Model (CDM) and the Climate and Forecast (CF) Conventions, which together provide a robust framework for representing scientific and geospatial datasets.

7.2.1. Common Data Model (CDM)

The CDM defines a generalised schema for representing array-based scientific datasets. The following constructs are reused directly within the unified model:

- **Dimensions** – Integer-valued, named axes that define the extents of data variables.
- **Coordinate Variables** – Variables that supply coordinate values along dimensions, establishing spatial or temporal context.
- **Data Variables** – Multidimensional arrays representing observed or simulated phenomena, associated with dimensions and coordinate variables.
- **Attributes** – Key-value metadata elements used to describe variables and datasets semantically.
- **Groups** – Optional hierarchical containers enabling logical organisation of resources and metadata.

The unified data model adopts these CDM components without modification excluding the user-defined types. Semantic interpretation remains consistent with the original CDM specification. GeoZarr structures are mapped to CDM constructs to ensure compatibility and clarity.

7.2.2. CF Conventions

The CF Conventions specify standardised metadata attributes and practices to describe spatiotemporal context within CDM-compliant datasets. These conventions support consistent interpretation of:

- Coordinate systems
- Grid mappings
- Physical units
- Standard variable naming

The unified data model supports CF-compliant metadata, including attributes such as `standard_name`, `units`, and `grid_mapping`. The unified data model does not prescribe CF compliance but enables it through permissive design. Partial adoption of CF attributes is supported, and non-compliant datasets may selectively adopt CF metadata as needed.

7.2.3. Standards-Based Extensions

To support additional capabilities, the model defines optional extension points referencing external OGC and community standards:

- **OGC Tile Matrix Set** – Facilitates the definition of multiscale grid hierarchies for raster overviews.
- **GDAL Geotransform** – Enables geospatial referencing through affine transformations and optional interpolation specifications.
- **STAC Metadata (Collection and Item)** – Provides linkage to SpatioTemporal Asset Catalogs for resource discovery and indexing.

These extensions are integrated in a modular fashion and do not alter the core semantics of the CDM or CF structures. Implementations may selectively adopt these extensions based on their application requirements.

7.3. Model Extension Points

The unified data model specifies a series of optional, standards-aligned extension points to support functionality beyond the base CDM and CF constructs. These extensions enhance applicability to Earth observation and spatial analysis use cases without imposing additional mandatory requirements.

Each extension is defined as an independent module. Implementation of any given extension does not necessitate support for others.

7.3.1. Multi-Resolution Overviews (OGC Tile Matrix Set)

Support for multi-resolution imagery is enabled via integration with the OGC Tile Matrix Set Standard:

- Tile matrix sets define spatial tiling schemes with consistent resolutions and coordinate reference systems across zoom levels.
- Overviews may be represented as separate Zarr arrays or groups, each aligned to a specific tile matrix level.
- Metadata includes identifiers for tile matrices, spatial resolution, and spatial alignment.

This approach aligns with the OGC API – Tiles and enables efficient access to large gridded datasets.

7.3.2. GeoTransform Metadata (GDAL Interpolation and Affine Transform)

Geospatial referencing can be further refined through the inclusion of metadata consistent with GDAL conventions:

- Affine transformation is specified via the `GeoTransform` attribute or equivalent structures.
- Interpolation methods may be declared to indicate sampling behaviour or sub-pixel alignment strategies.

This extension augments CF grid mappings by providing precise control over grid placement and coordinate transformations.

7.3.3. STAC Collection and Item Integration

To enable discovery of resources within the hierarchical structure of the data model, this Standard supports the inclusion of STAC metadata elements at appropriate locations within the group hierarchy.

A STAC extension consists of embedding or referencing STAC Collection and Item metadata within the data model:

- Each dataset resource MAY reference a corresponding STAC `Collection` or `Item` using an identifier or embedded object.
- STAC properties such as `datetime`, `bbox`, and `eo:bands` MAY be included in the metadata to enable spatial, temporal, and spectral filtering.
- The structure is compatible with external STAC APIs and metadata harvesting systems.

STAC integration is non-intrusive and modular. It does not impose changes on the internal organisation of datasets and MAY be adopted incrementally by implementations requiring catalogue-based discovery capabilities.

7.3.4. Modularity and Interoperability

Each extension point is specified independently. Implementations may advertise support for one or more extensions by declaring conformance to corresponding extension modules. This modularity facilitates incremental adoption, promotes reuse, and enhances interoperability across varied implementation environments.

7.4. Unified Model Structure

This clause defines the structural organisation of datasets conforming to the unified data model (UDM). It consolidates the foundational elements and optional extensions into a coherent architecture suitable for Zarr encoding, while remaining format-agnostic. The model establishes a modular and extensible framework that supports structured representation of multidimensional, geospatially-referenced resources.

The model represents datasets as abstract compositions of dimensions, coordinate variables, data variables, and associated metadata. This abstraction ensures that applications and services can reason about the content and semantics of a dataset without reliance on storage layout or specific serialisation.

7.4.1. Dataset Structure

A dataset conforming to the Unified Data Model (UDM) is structured as a hierarchy rooted at a top-level dataset entity. This design enables modularity and facilitates the representation of complex, multi-resolution, or thematically partitioned data collections.

Each dataset node comprises the following core components, aligned with the Unidata Common Data Model (CDM) and Climate and Forecast (CF) Conventions:

- **Dimensions** – Named, integer-valued axes defining the extent of data variables. Examples include `time`, `x`, `y`, and `band`.
- **Coordinate Variables** – Arrays that supply coordinate values along dimensions, providing spatial, temporal, or contextual referencing. These may be scalar or higher-dimensional, depending on the referencing scheme.
- **Data Variables** – Multidimensional arrays representing physical measurements or derived products. Defined over one or more dimensions, these variables are associated with coordinate variables and annotated with metadata.
- **Attributes** – Key-value pairs attached to variables or dataset components. Attributes convey semantic information such as units, standard names, and geospatial metadata.

The hierarchy is implemented through **groups**, which function as containers for variables, dimensions, and metadata. Groups may define local context while inheriting attributes from parent nodes. This supports the logical subdivision of datasets by theme, resolution, or processing stage, and enhances the clarity and reusability of complex geospatial structures.

The diagram below represents the structural layer of the unified data model, derived from the Unidata Common Data Model, which serves as the foundational framework for supporting all overlaying model layer.

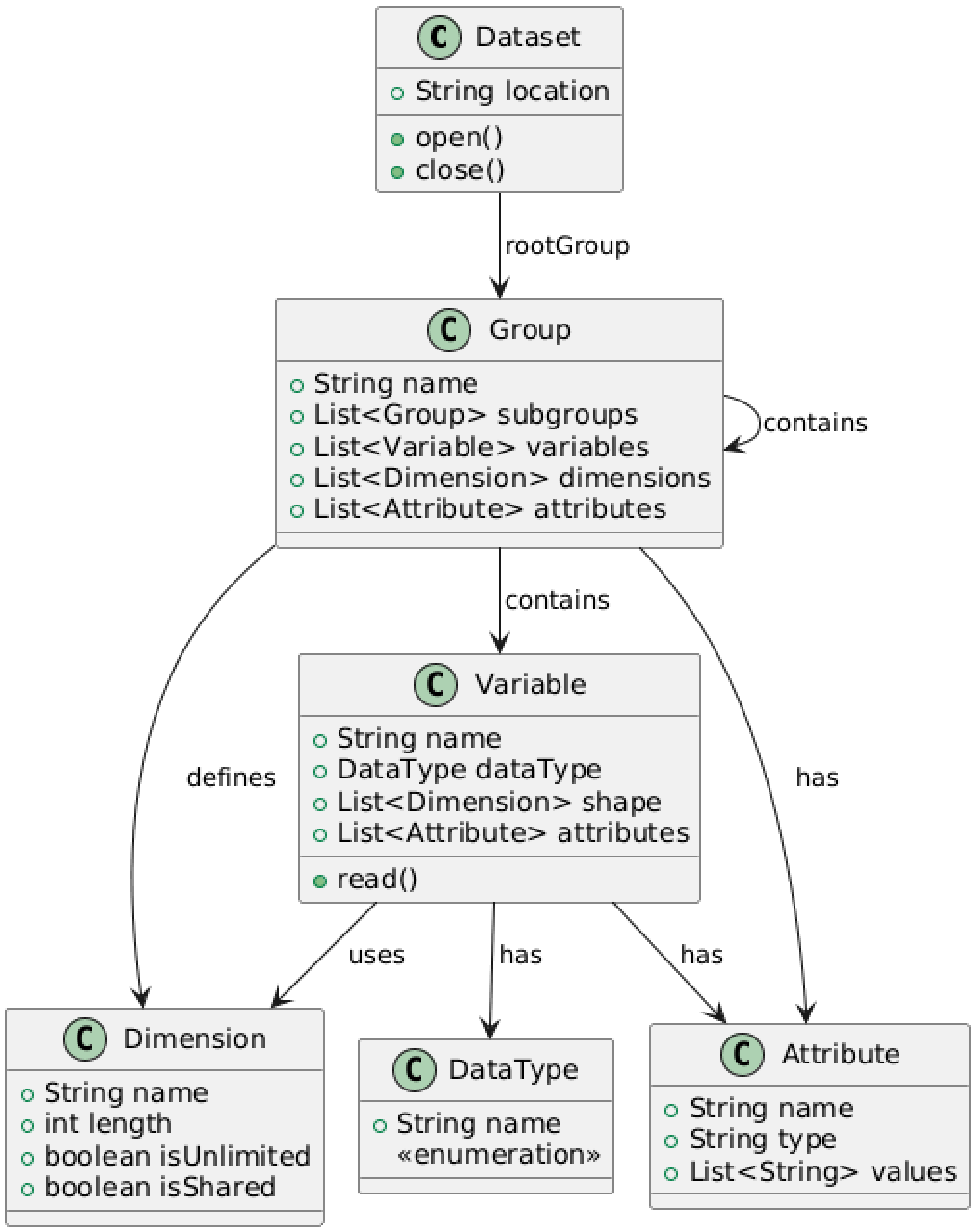

Figure 1 — Conformance-class model

Note that, conceptually, node within this hierarchy might be treated as a self-contained dataset.

7.4.2. Coordinate Referencing

Coordinate systems are defined using:

- **CF Conventions** – Including attributes such as `standard_name`, `units`, `axis`, and `grid_mapping` to express spatiotemporal semantics and coordinate system properties.

- **Affine Transformation Extensions** – Optional support for georeferencing via affine transforms and interpolation metadata (e.g., as defined in GDAL practices), providing enhanced flexibility for irregular grids and grid-aligned imagery.

The model accommodates both standard CF-compatible definitions and extended referencing mechanisms to support use cases that span scientific analysis and geospatial mapping.

7.4.3. Metadata Integration

Metadata may be declared at various levels within the model structure:

- **Global Metadata** – Attributes describing the dataset as a whole, including elements such as `title`, `summary`, and `license`.
- **Variable Metadata** – Attributes associated with individual data or coordinate variables, conveying descriptive or semantic information.
- **Extension Metadata** – Structured metadata linked to optional model extensions (e.g., multiscale tiling, catalogue references, geotransform properties).

All metadata follows harmonised naming and semantics consistent with the CDM and CF standards, enabling machine and human interpretability while supporting metadata exchange across diverse systems.

7.5. Overviews

7.5.1. Introduction

Overviews are downscaled representations of gridded data designed to optimise visualisation and scalable access to large datasets. Overviews or **multiscale pyramid** provide lower-resolution versions of the same variables, enabling rapid display, efficient zooming, and progressive data exploration. Multiple overview levels may exist, each representing the same data at a coarser spatial resolution.

The **Overviews** construct extends the Common Data Model (CDM) by defining a hierarchical organisation of groups and variables that represent data at multiple scales. The **OverviewSet** is self-described by attributes defined at the parent group level, which declare the relationships between its levels and ensure consistent, interoperable multiscale representation within the CDM framework.

7.5.2. Purpose and Scope

The **Overviews** extension enables scalable access to multidimensional gridded data, particularly for geospatial and remote sensing applications. It supports:

- Progressive rendering and visualisation at multiple resolutions
- Efficient data transfer for large datasets
- Multi-resolution analysis in analytical or cloud environments
- Consistent representation of raster and data cube structures across scales

This specification is format-agnostic and may be implemented in any CDM-compliant structure, regardless of physical encoding (e.g. Zarr, NetCDF, GeoTIFF) although the present specification specifically targets Zarr.

7.5.3. Conceptual Model

The Overviews construct defines a multiscale hierarchy applied to the variable group, i.e., the CDM group containing the data variables and their associated metadata, and optionally other related variables that share identical dimensions and coordinate systems.

Example: Typical CDM Group Structure without overviews

```
Group: variable_group/
├── variable1
├── variable2
├── aux_variable
├── coordinates1
├── coordinates2
└── Attributes
```

Listing 1

Each overview level provides a reduced-resolution representation of the same variables. This approach avoids redundancy by describing the hierarchy once for the entire group rather than for individual variables, ensuring consistency and concise metadata.

All overview levels are semantically equivalent, differing only in resolution, array extent, or sampling density. There is no requirement for a single base or reference level—each level may serve as an entry point depending on the application.

7.5.4. Model Components

The **Overviews** construct defines the conceptual elements used to represent multiscale data within the GeoZarr data model. It extends the existing CDM concept to support datasets provided at multiple spatial resolutions.

The construct introduces the following conceptual elements:

Table 2

ELEMENT	DEFINITION
`OverviewSet`	A **group** composed of multiple **OverviewLevels**, each containing equivalent variables defined over the same coordinate system and dimensions but sampled at different spatial resolutions. The **OverviewSet** defines the complete multiscale hierarchy.
`OverviewLevel`	A single resolution level within an **OverviewSet**. Each level replicates the structure and semantics of the others, differing only in resolution or extent.
`zoom_level`	An optional ordered identifier used to distinguish overview levels (e.g. `0`, `1`, `2` or symbolic identifiers). The ordering indicates relative resolution but does not imply dependency.

The OverviewSet retains the same structure as a nominal variable group, including the associated metadata and auxiliary variables, so that the multiscale hierarchy preserves the complete descriptive context of the original dataset

Example: Note: The native-resolution data **MAY** be stored directly in the **OverviewSet** group rather than in a dedicated **OverviewLevel** subgroup.
This layout is permitted for backward compatibility with existing datasets that were later augmented with multiscale metadata. However, it is **not recommended**, as it may lead to inconsistent hierarchies or interpretation issues in client applications expecting all resolution levels to be represented as explicit subgroups.

7.5.5. Structural Layout

The **Overviews** construct is expressed within the Common Data Model (CDM) framework, which represents datasets through **groups**, **variables**, and **attributes**.

An **OverviewSet** corresponds to a CDM **group** containing multiple **OverviewLevels**, each representing the same data variables at different spatial resolutions.

Within this structure:

- **Groups** define the hierarchical organisation of the multiscale data. The **OverviewSet** acts as the parent group, while each **OverviewLevel** is represented as a child group that contains variables with identical names, dimensions, and coordinate definitions. The **OverviewSet** group may also include auxiliary variables and metadata consistent with the structure of a nominal CDM group.

- **Variables** represent the same physical or derived quantities across resolutions. Each level contains the same set of data variables and coordinate variables (for example, `x` and `y`) that describe grid geometry at that resolution.

- **Attributes** describe both the dataset metadata and the relationships between overview levels. They may appear at the **OverviewSet** or **OverviewLevel** level and are used to define the structure and interpretation of the hierarchy.

The complete description of the hierarchy is provided by the `multiscales` property, an attribute of the **OverviewSet** group that lists the available overview levels, their identifiers, and any associated information such as resampling methods or grid references.

7.5.5.1. OverviewSet CDM-Based Representation

The following example illustrates the structural organisation of an **OverviewSet** using Common Data Model (CDM) constructs:

```
Group: reflectance/                     # OverviewSet (Group)
├── Attribute: multiscales              # Metadata describing the multiscale
hierarchy
├── Attribute: spatial_ref = "EPSG:32633"
├── Auxiliary Variable: quality_flag
├── Group: L0/                          # OverviewLevel (highest or nominal
resolution)
│      ├── Variable: b01
│      ├── Variable: b02
│      ├── Variable: b03
│      ├── Coordinate Variable: x
│      └── Coordinate Variable: y
├── Group: L1/                          # OverviewLevel (coarser resolution)
│      ├── Variable: b01
│      ├── Variable: b02
│      ├── Variable: b03
│      ├── Coordinate Variable: x
│      └── Coordinate Variable: y
└── Group: L2/                          # OverviewLevel (coarsest resolution)
       ├── Variable: b01
       ├── Variable: b02
       ├── Variable: b03
       ├── Coordinate Variable: x
       └── Coordinate Variable: y
```

Listing 2

In this representation:

- The **parent group** (`reflectant/`) corresponds to the **OverviewSet** and defines the common spatial, semantic, and organisational context for all levels.
- Each **child group** (`L0`, `L1`, `L2`) represents an **OverviewLevel**, implemented as a CDM **group** containing variables that share the same names, coordinate variables, and metadata conventions.
- **Variables** (`b01`, `b02`, etc.) represent equivalent physical quantities at different spatial resolutions.

7.5.6. OverviewSet Metadata

The `multiscales` property is an attribute of the **OverviewSet** group that formally defines the organisation of the multiscale hierarchy. It provides a structured description of all overview levels, their ordering, and the resampling or aggregation relationships between them.

The property SHALL be encoded as a structured object formally defined as a JSON Schema available at: Multiscales JSON Schema

It defines global attributes applying to the entire hierarchy and a `layout` array that lists all overview levels in order of resolution.

7.5.6.1. Multiscales Fields

Table 3

FIELD	DEFINITION
`version`	**Type:** string. Version identifier of the multiscales schema. This field SHALL be present to indicate the version of the schema used. Example: `"1.0"`
`resampling_method`	**Type:** string. (Optional) Default resampling or aggregation method applied across all levels. If omitted, resampling may be defined per level. Allowed values include: `"nearest"`, `"average"`, `"bilinear"`, `"cubic"`, `"cubic_spline"`, `"lanczos"`, `"mode"`, `"max"`, `"min"`, `"med"`, `"sum"`, `"q1"`, `"q3"`, `"rms"`, `"gauss"`. Default: `"nearest"`.
`tile_matrix_ref`	**Type:** string or object. (Optional) Reference to an external grid or tiling definition (e.g. an OGC Tile Matrix Set identifier or URI) that describes the spatial structure and scale relationships.
`layout`	**Type:** array of Overview Level Object. A mandatory array describing each **Overview Level** within the hierarchy, ordered from highest to lowest resolution. Each entry defines the group name and optional derivation information.
`variables`	**Type:** array of string. (Optional) List of variable names for which overviews are defined. If omitted, all variables in the parent group are assumed to have overviews. Example: `["b01", "b02", "b03"]`.

7.5.6.2. Overview Level Object

Each object in the `layout` array describes one **OverviewLevel** within the multiscale hierarchy. It defines a unique identifier for the level, its location within the dataset hierarchy, and optionally its derivation from another level.

Table 4

FIELD	DEFINITION
`id`	**Type:** string. Required unique identifier for this overview level. The identifier SHALL be stable within the dataset and MAY be used for reference in other metadata fields. Example: `"L0"`, `"L1"`, `"L2"`.
`path`	**Type:** string. (Optional) Logical path identifying the location of the overview level within the dataset hierarchy. If omitted, the level is assumed to be located as a **direct child group** of the **OverviewSet** and the `id` value SHALL be used as the default relative path. Example: `"L0"`, `"overviews/L2"`.
`derived_from`	**Type:** string. (Optional) Identifier of another overview level from which this level was derived. Used to express lineage or dependency relationships between levels. The value SHALL correspond to an existing `id` entry in the same `layout` array.
`cell_size`	**Type:** number or array of number. (Recommended) Resolution of the data at this level, expressed as the physical size of one cell in coordinate units (e.g. metres or degrees). This property allows clients such as map viewers to select the most appropriate overview level for a given display scale. If expressed as an array, the order SHALL match the spatial axes (e.g. `[x, y]`).
`factors`	**Type:** array of number. (Optional) Numeric decimation factors per dimension (e.g. `[2, 2]` for a 2× reduction in X and Y). Used to describe the scaling applied to generate this level from its source.
`resampling_method`	**Type:** string. (Optional) Resampling or aggregation method specific to this level. If not defined, the method specified in the root `multiscales.resampling_method` field applies.

7.5.6.3. Example Representation

Here is a JSON example that conforms to the **final `multiscales` schema**:

```
{
  "version": "1.0",
  "resampling_method": "average",
  "tile_matrix_ref": "OGC:WMT:1.0:WebMercatorQuad",
  "layout": [
    {
      "id": "L0",
      "path": "L0",
      "cell_size": [10.0, 10.0]
    },
    {
      "id": "L1",
      "path": "L1",
      "derived_from": "L0",
      "factors": [2, 2],
      "cell_size": [20.0, 20.0],
      "resampling_method": "average"
    },
    {
      "id": "L2",
      "path": "L2",
      "derived_from": "L1",
      "factors": [2, 2],
      "cell_size": [40.0, 40.0],
      "resampling_method": "average"
    }
  ]
}
```

Listing 3

Notes:

- Each `id` uniquely identifies an overview level.
- `path` points to the logical container for that level (may be omitted if it is a direct child of the `OverviewSet`).
- `derived_from` expresses lineage between levels.
- `factors` defines downscaling ratios.
- `resampling_method` can be defined per level or inherited from the global one.
- The `tile_matrix_ref` provide context for external referencing.

7.6. Conformance and Extensibility

The GeoZarr data model is designed with an open conformance approach to support a wide range of use cases and implementation contexts. Its core model is permissive, allowing partial implementations, while optional extensions and compliance profiles can define stricter requirements for interoperability.

7.6.1. Core Conformance

- Datasets conforming to the core model must:
  - Represent data using CDM-compatible constructs (dimensions, variables, attributes).
  - Follow attribute conventions where applicable.
  - Be parsable as valid Zarr with structured metadata following this specification.
- CF compliance is not mandatory but is recommended for semantic interoperability.

7.6.2. Extension Conformance

- Implementations may optionally support one or more extension modules:
  - Multi-resolution overviews (Tile Matrix Set)
  - GeoTransform metadata (GDAL)
  - STAC metadata integration
- Each extension defines its own requirement class with validation rules and expected metadata structures.
- Tools may advertise which extensions they support and validate datasets accordingly.

7.6.3. Conformance Classes

- Conformance Classes may be defined to specify required components and extensions for specific application domains (e.g., visualisation clients, EO archives, catalogue indexing).
- Conformance Classes enable selective validation without constraining the general model.

7.6.4. Extensibility Principles

- All extensions must preserve compatibility with the core model and avoid redefining existing CDM or CF semantics.
- New extensions should be documented with clear identifiers, schemas, and conformance criteria.
- The model encourages interoperability by allowing tools to interpret unknown extensions without failure.

This extensibility framework supports both minimum-viable use and high-fidelity metadata integration, enabling incremental adoption across the geospatial and scientific data communities.

7.7. Interoperability Considerations

Interoperability is a core objective of the GeoZarr unified data model. The model is designed to bridge diverse Earth observation and scientific data ecosystems by enabling structural and semantic compatibility with established formats and standards, while providing a forward-looking foundation for scalable, cloud-native workflows.

This section outlines the principles and mechanisms supporting interoperability across formats, tools, and communities.

7.7.1. Format Mapping and Alignment

The data model is explicitly aligned with foundational standards including the Unidata Common Data Model (CDM), the CF Conventions, and established practices in formats such as NetCDF and GeoTIFF. Where applicable, GeoZarr datasets may be derived from or transformed into these formats using consistent mappings.

- **NetCDF (classic and enhanced models)**:
  - GeoZarr shares a common conceptual structure with NetCDF via CDM.
  - Variables, dimensions, coordinate systems, and attributes follow directly mappable patterns.
  - Metadata expressed in CF conventions in NetCDF can be preserved in GeoZarr without loss of fidelity.
- **GeoTIFF**:
  - Raster-based datasets in GeoZarr can map to GeoTIFF by interpreting spatial referencing (via CF or GeoTransform) and band structures.
  - Overviews aligned to OGC Tile Matrix Sets may correspond to TIFF image pyramids.
  - Projection metadata and resolution information can be mapped via standard tags.

These mappings facilitate round-trip transformations and enable toolchains that consume or produce multiple formats without reengineering semantic models.

7.7.2. Semantic Interoperability

Semantic interoperability is supported through adherence to CF conventions, use of standardised attribute names (e.g., `standard_name`, `units`), and alignment with metadata vocabularies used in other ecosystems (e.g., STAC, EPSG codes, ISO 19115 keywords).

The model does not prescribe specific vocabularies beyond CF but encourages reuse and recognition of widely accepted descriptors to promote cross-domain understanding.

7.7.3. 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.

7.7.4. 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

GEOZARR CONFORMANCE CLASSES

8 GEOZARR CONFORMANCE CLASSES

Datasets can include many different types of data includes rasters, combinations—such as time, height, or wavelength—and can use either a projected or geographic coordinate system.

This Standard identifies conformance classes rg r offer clear, testable building blocks as a standardised approach for representing different data types when converting to the GeoZarr Unified Data Model (e.g. for encoding RGB bands from a GeoTIFF source).

This is a very preliminary draft. The content is primarily for demonstrating the purpose of the proposed sections.

9

UNIFIED DATA MODEL ENCODING FOR ZARR

9 UNIFIED DATA MODEL ENCODING FOR ZARR

This clause defines the encoding of the unified data model into the Zarr format. The encoding supports both Zarr Version 2 and Zarr Version 3.

This is a very preliminary draft. The content is primarily for demonstrating the purpose of the proposed sections.

9.1. Hierarchical Structure

A dataset conforming to the unified data model is represented as a hierarchical structure of groups, variables (arrays), dimensions, and metadata. The dataset is rooted in a **top-level group**, which may contain:

- Arrays representing coordinate or data variables
- Child groups for modular organisation, including logical sub-collections or resolution levels
- Metadata attributes at group and array levels

Each group adheres to a consistent structure, allowing recursive composition. This reflects the CDM's use of **groups** and is supported by both Zarr v2 and v3 with differing implementations.

Table 5

MODEL ELEMENT	ZARR V2 ENCODING	ZARR V3 ENCODING
Root Dataset	Directory with `.zgroup` and `.zattrs`	Directory with `zarr.json`, with `node_type: group`
Child Group	Subdirectory with `.zgroup` and `.zattrs`	Subdirectory with `zarr.json`, with `node_type: group`
Array	Subdirectory with `.zarray` and `.zattrs`	Subdirectory with `zarr.json`, with `node_type: array`
Attributes	`.zattrs` file	`attributes` field in `zarr.json`

Zarr v3 requires `zarr_format: 3` and stores all metadata (including user-defined attributes) in the `zarr.json` document. Each node includes a `node_type` field: either `"group"` or `"array"`.

9.2. Dimensions

Dimensions define the axes along which variables are indexed.

- In Zarr v2, dimensions are inferred from array shape and declared in `_ARRAY_DIMENSIONS` within `.zattrs`.
- In Zarr v3, dimensions are stored using the `dimension_names` field in `zarr.json`.

Example for a 2D array with dimension names `["lat", "lon"]`:

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

Listing 4

9.3. Coordinate Variables

Coordinate variables (excluding GeoTransform Coordinates) define the geospatial or temporal context of data. They are represented as named arrays with metadata attributes.

Coordinate variables are represented as named 1D arrays aligned with corresponding dimensions.

Table 6

FEATURE	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`
CF Metadata	`standard_name`, `units`, `axis` in `.zattrs`	Under `attributes` in `zarr.json`

Example `zarr.json` for a coordinate array:

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

Listing 5

9.4. Data Variables

Data variables represent measured or derived quantities. They are stored as multidimensional arrays with metadata attributes.

Table 7

FEATURE	ZARR V2	ZARR V3
Storage	Multidimensional array with `.zarray` and `.zattrs`	Same structure; v3 supports additional chunk storage formats
Dimension Binding	`_ARRAY_DIMENSIONS` in `.zattrs`	`dimension_names` in `zarr.json`
CF Metadata	`standard_name`, `units`, `long_name`, `_FillValue`, etc.	Same as v2; v3 may support typed attributes

Example:

```
{
  "_ARRAY_DIMENSIONS": ["time", "lat", "lon"],
  "standard_name": "air_temperature",
  "units": "K",
  "long_name": "Surface air temperature",
  "_FillValue": -9999.0
}
```

Listing 6

9.5. Global Metadata

Metadata associated with the dataset as a whole is stored at the root group level.

Table 8

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

Example Zarr v3 root `zarr.json`:

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

Listing 7

9.6. Variables Metadata

All metadata attributes (for groups, coordinates variables and data variables) are recommended to conform to CF naming and typing conventions. Supported attributes include:

- `standard_name`, `units`, `axis`, `grid_mapping` (CF)
- `_FillValue`, `scale_factor`, `add_offset`
- `long_name`, `missing_value`

In all cases:

- Attribute names are case-sensitive and encoded as UTF-8 strings
- Values shall conform to JSON-compatible types (string, number, boolean, array)

9.7. Encoding of Multiscale Overviews in Zarr

This clause specifies how Overviews should be encoded in Zarr-based datasets conforming to the GeoZarr data model.

The **Overviews** construct follows the same encoding principles defined for the Common Data Model (CDM) when represented in Zarr. Because overviews are defined purely at the CDM level through groups, variables, and attributes, no Zarr-specific structural extensions are introduced.

Each **OverviewSet** is encoded as a Zarr **group**, containing multiple **OverviewLevel** subgroups. The parent group includes the `multiscales` attribute, which declares the hierarchy and relationships between levels. Each **OverviewLevel** is implemented as a standard Zarr group containing variables (arrays) and coordinate variables, encoded identically to other CDM variables.

9.7.1. Relationship to Core CDM Encoding

The encoding of overviews reuses the same mapping rules established in the core data model encoding:

- **Groups** map to Zarr directories with `.zgroup`/`.zattrs` (Zarr v2) or `zarr.json` (Zarr v3, with `"node_type": "group"`).
- **Variables** map to Zarr arrays with `.zarray` and `.zattrs` (Zarr v2) or `"node_type": "array"` entries in `zarr.json` (Zarr v3).
- **Attributes** (including `multiscales`) are stored in `.zattrs` (Zarr v2) or under the `"attributes"` field in `zarr.json` (Zarr v3).

9.7.2. Example Encoding (Zarr v3)

```
{
  "zarr_format": 3,
  "node_type": "group",
  "attributes": {
    "multiscales": {
      "version": "1.0",
      "resampling_method": "average",
      "tile_matrix_ref": "OGC:WMT:1.0:WebMercatorQuad",
      "layout": [
        {
          "id": "L0",
          "path": "L0",
          "cell_size": [10.0, 10.0]
        },
        {
          "id": "L1",
          "path": "L1",
          "derived_from": "L0",
          "factors": [2, 2],
          "cell_size": [20.0, 20.0],
          "resampling_method": "average"
        },
        {
          "id": "L2",
          "path": "L2",
          "derived_from": "L1",
          "factors": [2, 2],
          "cell_size": [40.0, 40.0],
          "resampling_method": "average"
        }
      ]
    },
    "spatial_ref": "EPSG:32633"
  },
  "metadata": {
    "title": "Reflectance Multiscale Example",
    "summary": "Example of a multiscale dataset encoded using the GeoZarr
Overviews extension."
  }
}
```

Listing 8

Child groups represent the overview levels:

```
reflectance/
├── zarr.json                        # OverviewSet metadata (includes
"multiscales")
├── L0/
│   ├── zarr.json                    # Highest or nominal resolution
│   ├── b01/
│   ├── b02/
│   ├── x/
│   └── y/
├── L1/
│   ├── zarr.json
│   ├── b01/
│   ├── b02/
│   ├── x/
│   └── y/
└── L2/
├── zarr.json
├── b01/
├── b02/
├── x/
└── y/
```

Listing 9

10

UNIFIED DATA MODEL ENCODING FOR GEOTIFF

10 UNIFIED DATA MODEL ENCODING FOR GEOTIFF

This is a very preliminary draft. The content is primarily for demonstrating the purpose of the proposed sections.

A

ANNEX A (INFORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

A ANNEX A (INFORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

NOTE: Ensure that there is a conformance class for each requirements class and a test for each requirement (identified by requirement name and number)

A.1. Conformance Class A

Example

label	http://www.opengis.net/spec/name-of-standard/1.0/conf/example1
subject	Requirements Class “example1”
classification	Target Type:Web API

A.1.1. Example 1

ABSTRACT TEST A.1	
SUBJECT	/req/req-class-a/req-name-1
LABEL	/conf/core/api-definition-op
TEST PURPOSE	Validate that the API Definition document can be retrieved from the expected location.
TEST METHOD	1. Construct a path for the API Definition document that ends with `/api`. 2. Issue a HTTP GET request on that path 3. Validate the contents of the returned document using test /conf/core/api-definition-success.

A.1.2. Example 2

ABSTRACT TEST A.2	
SUBJECT	/req/req-class-a/req-name-2
LABEL	/conf/core/http
TEST PURPOSE	Validate that the resource paths advertised through the API conform with HTTP 1.1 and, where appropriate, TLS.
TEST METHOD	1. All compliance tests SHALL be configured to use the HTTP 1.1 protocol exclusively. 2. For APIs which support HTTPS, all compliance tests SHALL be configured to use HTTP over TLS (RFC 2818) with their HTTP 1.1 protocol.

B

ANNEX B (INFORMATIVE) TITLE

B

ANNEX B (INFORMATIVE) TITLE

NOTE: Place other Annex material in sequential annexes beginning with “B” and leave final two annexes for the Revision History and Bibliography

C

ANNEX C (INFORMATIVE) REVISION HISTORY

C

ANNEX C (INFORMATIVE) REVISION HISTORY

Table C.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2016-04-28	0.1	G. Editor	all	initial version

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] *OGC: OGC Testbed 12 Annex B: Architecture* (2015).