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

License Agreement

Use of this document is subject to the license agreement at https://www.ogc.org/license

Suggested additions, changes and comments on this document are welcome and encouraged. Such suggestions may be submitted using the online change request form on OGC web site: http://ogc.standardstracker.org/

Copyright notice

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

I ABSTRACT

<Insert Abstract Text here>

II KEYWORDS

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

ogcdoc, OGC document, API, openapi, html

PREFACE

NOTE: Insert Preface Text here. Give OGC specific commentary: describe the technical content, reason for document, history of the document and precursors, and plans for future work.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

IV SECURITY CONSIDERATIONS

No security considerations have been made for this Standard.

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:

Name	Affiliation

VII CONTRIBUTORS

Additional contributors to this Standard include the following:

Individual name(s), Organization

PREFACE

PREFACE

NOTE: Insert Preface Text here. Give OGC specific commentary: describe the technical content, reason for document, history of the document and precursors, and plans for future work. > Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

1 SCOPE

1 SCOPE

NOTE: Insert Scope text here. Give the subject of the document and the aspects of that scope covered by the document.

2

CONFORMANCE

2 CONFORMANCE

This standard defines XXXX.

Requirements for N standardization target types are considered:

- AAAA
- BBBB

Conformance with this standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site.

In order to conform to this OGC® interface standard, a software implementation shall choose to implement:

- Any one of the conformance levels specified in Annex A (normative).
- Any one of the Distributed Computing Platform profiles specified in Annexes TBD through TBD (normative).

All requirements-classes and conformance-classes described in this document are owned by the standard(s) identified.

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.

Identification of Common Molecular Subsequences. Smith, T.F., Waterman, M.S., J. Mol. Biol. 147, 195–197 (1981)

4 TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

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.

This document uses the terms defined in Sub-clause 5.3 of [OGC06-121r9], 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 standard.

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

4.1. example term

term used for exemplary purposes

Note 1 to entry: An example note.

Example Here’s an example of an example term.

[**SOURCE:**]

5

CONVENTIONS

5 CONVENTIONS

This sections provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Identifiers

The normative provisions in this standard are denoted by the URI

`http://www.opengis.net/spec/{standard}/{m.n}`

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

6

CLAUSES NOT CONTAINING NORMATIVE MATERIAL

6 CLAUSES NOT CONTAINING NORMATIVE MATERIAL

Paragraph

6.1. Clauses not containing normative material sub-clause 1

Paragraph

6.2. Clauses not containing normative material sub-clause 2

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.

```
1      @startuml CDM_DAL_Object_Model
2
3      class Dataset {
```

```
        + String location
        + open()
        + close()
    }

    class Group {
        + String name
        + List<Group> subgroups
        + List<Variable> variables
        + List<Dimension> dimensions
        + List<Attribute> attributes
    }

    class Dimension {
        + String name
        + int length
        + boolean isUnlimited
        + boolean isShared
    }

    class Variable {
        + String name
        + DataType dataType
        + List<Dimension> shape
        + List<Attribute> attributes
        + read()
    }

    class DataType {
        + String name
        <<enumeration>>
    }

    class Attribute {
        + String name
        + String type
        + List<String> values
    }

    Dataset --> Group : rootGroup
    Group --> Group : contains >
    Group --> Variable : contains >
    Group --> Dimension : defines >
    Group --> Attribute : has >
    Variable --> Dimension : uses >
    Variable --> DataType : has >
    Variable --> Attribute : has >
    @enduml
```

Listing 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.

7.4.4. Overviews

The **Overviews** construct defines a formal, interoperable abstraction for multiscale gridded data. It ensures structural consistency across zoom levels and provides a semantic model for integration with tiled representations such as GeoTIFF overviews, OGC API – Tiles, and STAC Tiled Assets.

7.4.4.1. Purpose

The **Overviews** construct provides a general mechanism for associating a single logical data variable with a collection of resampled representations, referred to as **zoom levels**. Each zoom level holds a reduced-resolution version of the original variable, with progressively decreasing spatial resolution from the base (highest detail) to the coarsest level.

Overviews enable:

- Fast access to summary representations for visualisation
- Progressive transmission and downsampling

- Multi-resolution analytics and adaptive processing

7.4.4.2. Conceptual Structure

An **Overviews** construct is defined as a **hierarchical set of multiscale representations** of one or more data variables. It comprises the following components:

Base Variable	The original, highest-resolution variable to which the overview hierarchy is anchored. It is defined using the standard `DataVariable` structure in the model.
Overview Levels	A sequence of variables representing the same logical quantity as the base variable, but sampled at coarser spatial resolutions.
Zoom Level Identifier	A unique identifier associated with each level, ordered from finest (e.g. `"0"`) to coarsest resolution (e.g. `"N"`).
Tile Grid Definition	A mapping that associates each zoom level with a spatial tiling layout, defined in alignment with a `TileMatrixSet`.
Spatial Alignment	Each overview variable MUST be spatially aligned with the base variable using a consistent coordinate reference system and compatible axis orientation.
Resampling Method	A declared method indicating the technique used to derive coarser levels from the base variable (e.g. `nearest`, `average`, `cubic`).

7.4.4.3. Model Components

The **Overviews** construct is represented in the unified data model using the following logical elements:

Table 1

ELEMENT	DEFINITION
`OverviewSet`	A logical grouping of variables at multiple zoom levels associated with a single base variable.
`OverviewLevel`	A single resampled variable at a specific resolution, identified by a zoom level string.
`TileMatrixSetRef`	A reference to the tile grid specification applied across all overview levels. May refer to a well-known identifier, a URI, or an inline object.
`TileMatrixLimits`	(Optional) Constraints on the tile coverage per zoom level.
`resampling_method`	A string indicating the uniform method used to downsample data across all levels.

All overview levels MUST preserve:

- The data variable's semantic identity (`standard_name`, `units`, etc.)
- The coordinate reference system
- The axis order and dimension semantics

Only the resolution and extent (through tiling and shape) may differ across levels.

7.4.4.4. Relationship to Tile Matrix Set

The **Overviews** construct is structurally aligned with the OGC Tile Matrix Set concept. Each zoom level is mapped to a `TileMatrix`, and the chunk layout for the corresponding data variable SHALL match the tile grid's `tileWidth` and `tileHeight`.

The `OverviewSet` MAY constrain tile matrix limits using `TileMatrixSetLimits`, which restrict tile indices to actual data coverage, consistent with the spatial extent of the overview variable.

7.4.4.5. Usage Context

The **Overviews** construct is applicable to any gridded data variable with at least two spatial dimensions. It is primarily designed for:

- Raster imagery (e.g. reflectance, temperature)
- Data cubes with spatial slices (e.g. time-series of spatial grids)
- Multi-band products with consistent spatial structure across levels

The structure may be extended for N-dimensional datasets in future revisions, provided that two spatial axes can be unambiguously identified.

7.5. 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.

7.5.1. 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.

7.5.2. 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.

7.5.3. 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.

7.5.4. 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.

7.6. 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.

7.6.1. 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.

7.6.2. 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.

7.6.3. 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.

7.6.4. 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 2

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`
Metadata Key	`.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 2

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 3

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 3

9.4. Data Variables

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

Table 4

FEATURE	ZARR V2	ZARR V3
Storage	Multidimensional array with `.zarray` and `.zattrs`	Same structure; v3 supports additional chunk storage formats
Dimension Association	`_ARRAY_DIMENSIONS` attribute	Same as v2
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 4

9.5. Global Metadata

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

Table 5

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 5

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 multiscale tiling (also known as overviews or pyramids) is encoded in Zarr-based datasets 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.

Multiscale datasets are composed of a set of Zarr groups representing multiple zoom levels. Each level stores coarser-resolution resampled versions of the original data variables.

9.7.1. Hierarchical Layout

Each zoom level SHALL be represented as a Zarr group, identified by the Tile Matrix identifier (e.g., `"0"`, `"1"`, `"2"`). These groups SHALL be organised hierarchically under a common multiscale root group. Each zoom-level group SHALL contain the complete set of variables (Zarr arrays) corresponding to that resolution.

Table 6

STRUCTURE	ZARR V2	ZARR V3
Zoom level groups	Subdirectories with `.zgroup` and `.zattrs`	Subdirectories with `zarr.json`, `node_type: group`
Variables at each level	Zarr arrays (`.zarray`, `.zattrs`) in each group	Zarr arrays (`zarr.json`, `node_type: array`) in each group
Global metadata	`multiscales` defined in parent `.zattrs`	`multiscales` defined in parent group `zarr.json` under `attributes`

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

9.7.2. Metadata Encoding

Multiscale metadata SHALL be defined using a `multiscales` attribute located in the parent group of the zoom levels. 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

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

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

Listing 6

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

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

Listing 7

9.7.3. 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.

9.7.4. 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)

9.7.5. 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 8

9.7.6. 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.

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)

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

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

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

NOTE: The TC has approved Springer LNCS as the official document citation type.

Springer LNCS is widely used in technical and computer science journals and other publications

– Actual References:

[n] Journal: Author Surname, A.: Title. Publication Title. Volume number, Issue number, Pages Used (Year Published)

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