ISO/DIS 19123-2:2025(en)

ISO/TC 211/WG 6

Secretariat: SIS

Date: 2025-08-20

Geographic information — Schema for coverage geometry and functions — Part 2: Coverage implementation schema

ISO copyright office

CP 401 • Ch. de Blandonnet 8

CH-1214 Vernier, Geneva

Phone: +41 22 749 01 11

Email: copyright@iso.org

Website: www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 211, *Geographic information/Geomatics*, in collaboration with the European Committee for Standardization (CEN) Technical Committee CEN/TC 287, *Geographic Information,* in accordance with the Agreement on technical cooperation between ISO and CEN (Vienna Agreement), under participation of the IEEE GRSS Earth Science Informatics (ESI) Technical Committee, and derived from the Open Geospatial Consortium (OGC) standards CIS 1.0 and CIS 1.1 with permission.

This second edition cancels and replaces the first edition (ISO 19123-2:2018), which has been technically revised. The main changes of this document over its predecessor version ISO 19123-2:2018 are as follows:

— The document is adjusted to the structure established in ISO 19123-1. Among others, a clear separation of logical level (UML structures) and physical level (encodings, such as XML) is established.

— Use of terminology is adjusted to align with ISO 19123-1.

— Coverage type `GeneralGridCoverage` is integrated from OGC CIS 1.1[18] as an additional coverage structure which generalizes and simplifies grid coverage modelling. It adds comprehensive definitions for all possible types of irregular grids, including `RectifiedGridCoverage` and `ReferenceableGridCoverage` as special cases.

— `RectifiedGridCoverage` and `ReferenceableGridCoverage` have been moved into a (normative) annex as their specification follows a different logic which does not easily fit into the structuring given by the Coverage Fundamentals (ISO 19123-1). They form a legacy and will be deprecated in the next edition of this document as `GeneralGridCoverage` covers these cases while simpler in structure.

— Editorial changes have been made to the structure and nomenclature in order to conform to the most recent edition of the ISO/IEC Directives.

— The UML schema is updated to reflect the above updates.

— The XML coverage encoding is complemented with a structurally equivalent JSON encoding, based on modern JSON schema design patterns.

— The XML and JSON schemas of the coverage range type, which relies on the SWE Common DataRecord, have been copied verbatim from SWE Common into the coverage schema to make this document more self-contained and easier to read.

Note The RDF encoding of OGC CIS 1.1 has not been included at this time. It will possibly be added at a later time.

Technically, this document implements the following improvements over the previous version 19123-2:2018:

— More general grid identifiers (with punctuation, national character sets, etc.).

— Mixed regular and non-regular grids.

— Added support for non-regularly gridded sensor models.

— Clear regulation for interpolation methods associated with grid coverages, thereby also clarifying a long-standing confusion between discrete and continuous grid coverages.

— Distinction between grid dimension and CRS dimension.

— Introduction of `EnvelopeByAxis`, an envelope type which allows for a convenient handling of any type of coordinates.

— Partitioned (“tiled”) coverages, allowing – among others – “interleaved representations” of coverages and datacubes tiled for efficient subsetting.

— Removal of a namespace ambiguity in `ReferenceableGridCoverage` (resolved by introduction of `GeneralGridCoverage`).

— Some GML schema definitions whose generality complicates coverage understanding unnecessarily have been extracted and condensed into the pertaining XML schema. As a consequence, the XML encoding of this document is now a compact, freestanding definition, rather than a GML application schema. Nevertheless, by keeping with the coverage types inherited from the previous edition, ISO 19123-2:2018, it is possible for implementers to remain in the realm of a GML application schema.

A list of all the parts in the ISO 19123-x series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at iso.org/members.html.

Introduction

This document specifies an interoperable, conformance-testable information schema for coverages. As defined in ISO 19123-1[10] (which is equivalent to OGC Abstract Topic 6.1), coverages serve as digital representations of space-time varying phenomena, corresponding to the notion of a “field” in physics. Such coverages can be discrete or continuous. Common examples include 1-D time series, 2-D imagery, 3-D x/y/t image time series and x/y/z geophysical voxel models, as well as 4-D x/y/z/t atmospheric and ocean data. Coverages are independent from service definitions and, therefore, can be accessed through a variety of web based service types, such as the OGC Web Coverage Service (WCS) Standard,[6] and through service instantiations realizing ISO 19123-3.

This document is a compliant standardization target of ISO 19123-1:2023 relying on its concepts, terms, definitions, and interfaces to establish a logical schema (via UML) implementing the interfaces defined there. Additionally, this document defines related physical coverage schemas (via format encodings) for the single logical schema. Thus, ISO 19123-1 and this document together establish an abstraction hierarchy:

— conceptual level: ISO 19123-1 defining abstract interfaces;

— logical level: Clauses 5 – 10 of this document, defining data as object classes with attributes;

— physical level: Clauses 11 and 12 of this document, plus further separate coverage encoding standards defined outside this document, defining the mapping of the logical-level data to byte streams (GeoTIFF, netCDF, JPEG2000, etc.).

Geographic information — Schema for coverage geometry and functions — Part 2: Coverage implementation schema

1 Scope

This document specifies an implementable, conformance-testable coverage structure based on the abstract schema for coverages defined in the ISO 19123-1 coverage fundamentals. This document defines a concrete data structure that is suitable for encoding in many formats.

2 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.

ISO 19123-1:2023, *Geographic information — Schema for coverage geometry and functions — Part 1: Fundamentals*

ISO 19136-1:2020, *Geographic information — Geography Markup Language (GML) — Part 1: Fundamentals*

OGC 08-094r1, OGC® SWE *Common Data Model Encoding Standard, version 2.0*

OGC 24-014, OGC® SWE *Common Data Model Encoding Standard, version 3.0*

3 Terms, definitions, and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 19123-1:2023 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at https://www.electropedia.org/

— ISO Online browsing platform: available at https://www.iso.org/obp

3.1.1
coverage CRS
coverage Coordinate Reference System
<coverage> coordinate reference system (CRS) in which all coordinates in a coverage domain are expressed

Note 1 to entry: Sometimes a coverage's CRS is also referred to as the coverage's native CRS to express that this is the CRS to which all the coverage's location data refer.

Note 2 to entry: The domain <coverage> has been added to the definition.

3.1.2
displaced grid
<coverage> grid whose direct positions are topologically aligned to a grid, but whose geometric positions can vary arbitrarily

3.1.3
irregular grid
<coverage> grid whose direct positions have individual distances along each grid's axis

3.1.4
partition
<coverage> separately stored coverage acting, by being referenced in another coverage, as one of its components

3.1.5
regular grid
<coverage> grid whose direct positions have a constant distance along each grid's axis

3.1.6
transformation grid
<coverage> grid whose direct positions are given by a transformation

Note 1 to entry: This definition is adapted from OGC 08-094r1.

Note 2 to entry: The transformation algorithm described in this definition is not within the scope of this document.

3.2 Abbreviated terms

CIS	Coverage Implementation Schema
CRS	Coordinate Reference System
EPSG	European Petroleum Survey Group
GeoTIFF	Geo Tagged Image File Format
GML	Geography Markup Language
JSON	JavaScript Object Notation
netCDF	network Common Data Format
OGC	Open Geospatial Consortium
RDF	Resource Description Framework
SWE	Sensor Web Enablement
TIN	Triangulated Irregular Network
UoM	Unit of Measure
UML	Unified Modeling Language
WCS	Web Coverage Service
WCPS	Web Coverage Processing Service

4 Conformance

4.1 Notation

Schemas are presented using the Unified Modeling Language (UML) as defined in ISO 19103.[40]

4.2 Interoperability and conformance testing

The term “coverage”, together with related terms, is used as defined in ISO 19123-1:2023.[10] This Coverage Implementation Schema standard implements Clauses 5 through 10 of ISO 19123-1:2023 (ISO 19123-1:2023, Annex D is implemented by normative Annex B of this document). In other words, the data structures defined in the UML schema form an implementation of the interfaces established in ISO 19123-1:2023.

This document defines testable conformance classes which correspond to the requirements in Clause 5 onwards. These conformance classes, together with the corresponding conformance tests, are described in Annex A. Any implementation claiming conformance with this document must conform to the abstract conformance class “coverage” and, in addition, at least one of the conformance classes “xml”, “json”, “multipart”.

4.3 Organization

The coverage schema is organized into the packages shown in Figure 1. Each package establishes one requirements class. Figure 1 show the requirements class dependencies depicted as a UML package diagram; each package represents one class, the *depends-on* relationship represents the requirements class dependency relationship. Packages have been grouped along practical implementation considerations, in particular to maximize modularity (Figure 1 and Table 1):

- The core class “coverage” (in red). This is the only abstract class – it establishes the basic framework, while the concrete conformance classes listed below define how concrete coverage instances can be built.

- The grid coverage classes (in green):

  - Class “grid-regular” establishes multi-dimensional unreferenced and regular referenced grids; in particular, `GridCoverage` and `RectifiedGridCoverage` are provided here for backwards compatibility with version 1.0 of this standard.

  - Class “grid-irregular” establishes multi-dimensional irregular referenced grids.

  - Class “grid-transformation” establishes multi-dimensional referenced grids defined by algorithmic transformations.

- The non-gridded coverage classes (in blue):

  - Class “pointcloud” establishes point clouds, a class kept separately as it frequently appears standalone in practice.

  - Class “mesh” establishes general multi-dimensional geometric meshes with curves, surfaces, and solids.

- The format encoding classes (in yellow):

  - Class “xml-coverage” establishes XML encoding of coverages.

— Class “json-coverage” establishes JSON encoding of coverages.

— Class “multipart” establishes a multipart encoding of coverages.

— Class “partitioning” (in grey) establishes coverages composed from several sub-coverages.

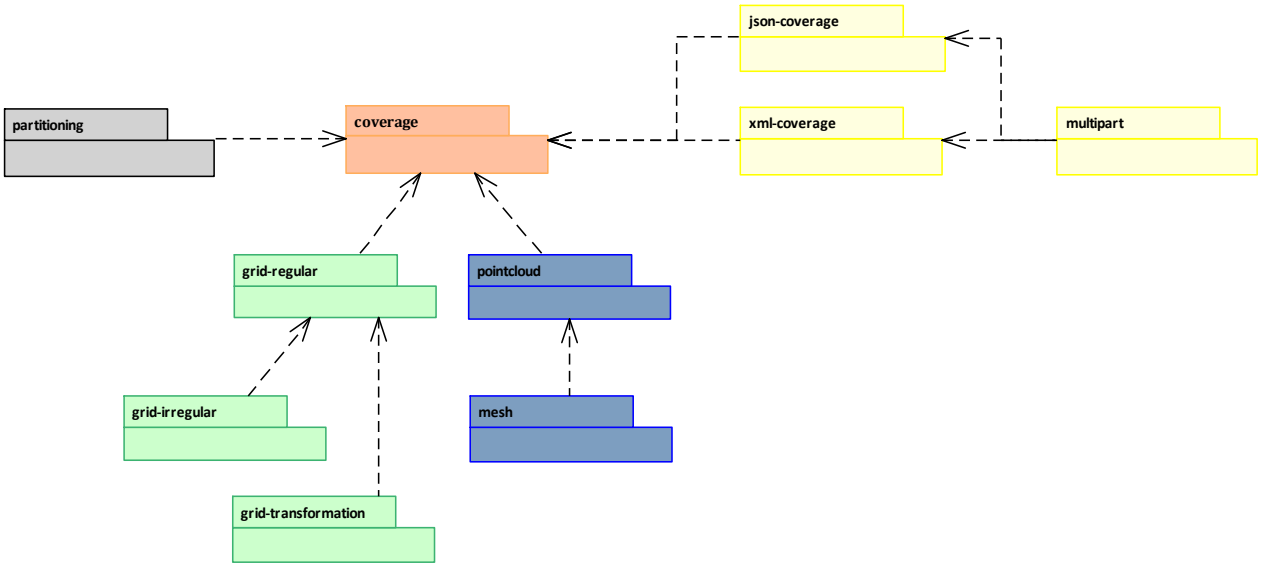

Figure 1 — The Coverage class hierarchy as UML package diagram

Requirement 1: https://standards.isotc211.org/19123/-/2/2/req/coverage/conformance

An implementation of this CIS standard, in order to be compliant, shall conform to:

— be an implementation of ISO 19123-1

— the core conformance class coverage plus

— at least one of the grid-regular, grid-irregular, grid-transformation, pointcloud, and mesh conformance classes plus

— at least one of the encoding conformance classes xml-coverage and json-coverage plus
— the conformance tests specified in Annex A.

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

The URIs in this standard have a stem of https://standards.isotc211.org/19123/-2/2/ which corresponds to the OGC URI stem https://www.opengis.net/spec/CIS/1.2/, except that the numbering (such as of requirements) has changed sometimes.

Table 1 — Package (conformance class) URIs established in this standard

Class	Description and URI
coverage	General, abstract coverage class, implementing ISO 19123-1:2023 https://standards.isotc211.org/19123/-2/2/conf/coverage
pointcloud	Coverage specialization for Multi-Point Coverages (point clouds) https://standards.isotc211.org/19123/-2/2/conf/pointcloud
grid-regular	Coverage specialization for regular Grid Coverages

Class	Description and URI
	https://standards.isotc211.org/19123/-2/2/conf/grid-regular
grid-irregular	Coverage specialization for irregular Grid Coverages https://standards.isotc211.org/19123/-2/2/conf/grid-irregular
grid-transformation	Coverage specialization for transformation Grid Coverages https://standards.isotc211.org/19123/-2/2/conf/grid-transformation
mesh	Coverage specialization for curve, surface, and solid coverages https://standards.isotc211.org/19123/-2/2/conf/mesh
partitioning	Partitioned representation of coverages https://standards.isotc211.org/19123/-2/2/conf/partitioning
xml-coverage	Coverage encoding in XML https://standards.isotc211.org/19123/-2/2/conf/xml-coverage
json-coverage	Coverage encoding in JSON https://standards.isotc211.org/19123/-2/2/conf/json-coverage
multipart	Multi-part representation of coverages https://standards.isotc211.org/19123/-2/2/conf/multipart

This document consists of the UML diagrams and textual requirements classes established in this document as well as an external file bundle consisting of the corresponding schema files, plus example coverage files.

The name and contact information of the maintenance agency for this document can be found at www.iso.org/maintenance_agencies.

5 Coverages

5.1 Overview

Conformance class *coverage* lays the foundation for the Coverage Implementation Schema. It is the abstract core class, meaning it does not allow creating coverage instances itself, but rather provides the fundament for the further classes which define various specializations of coverage instances.

The following ISO 19123-1:2023[10] coverages are sorted along the topological dimension of the elements they contain (Figure 2), visible in their respective domain set structure:

— 0-D point sets, known as Multi-Point Coverages for irregular point agglomerations (see Clause 6) and General Grid Coverages for points sitting on some regular or irregular grid (see Clause 7);

— 1-D curve bundles, known as Multi-Curve Coverages (see Clause 8);

— 2-D surface bundles, known as Multi-Surface Coverages (see Clause 9);

— 3-D solid bundles, known as Multi-Solid Coverages (see Subclause 10).

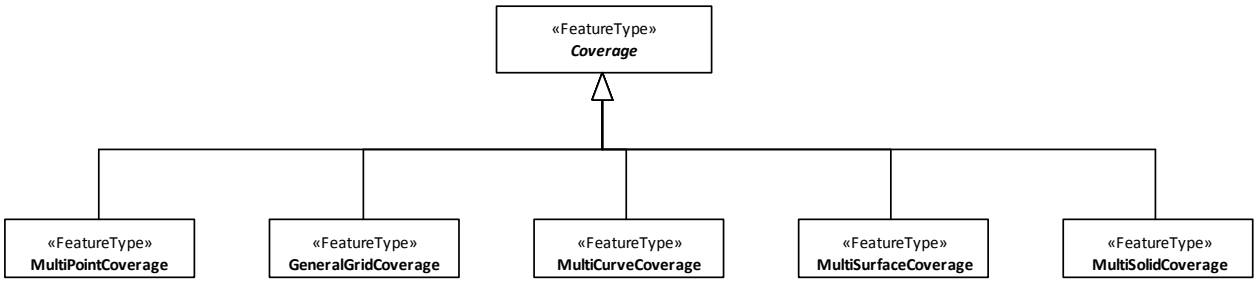

Figure 2 — `Coverage` subtypes defined in CIS (overview)

Requirement 2: https://standards.isotc211.org/19123/-/2/2/req/coverage/subtypes

A coverage shall be one of: `MultiPointCoverage`, `GeneralGridCoverage`, `MultiCurveCoverage`, `MultiSurfaceCoverage`, `MultiSolidCoverage`.

NOTE This structuring, adopted from ISO 19123-1:2023, is different from the previous edition of this standard, 19123-2:2018.

5.2 General coverage structure

A coverage consists of the following main components (Figure 3):

— *Domain set*: "where are values available?" Coordinate positions for which values are stored in the coverage are called *direct positions*.

— *Range set*: "what is the value at a particular position?" (often referred to as "pixels" or "voxels"). Such values can be atomic (such as in grayscale images) or record structures (such as in color or hyperspectral images). Record components are known as bands, channels, and variables in different disciplines.

— *Range type*: "what do these values mean?" Such a type often consists of one or more fields (also referred to as bands or channels or variables – not to be confused with the physics field a coverage represents). For this description of the semantics, coverages in this document make use of OGC SWE Common[16].

— *Metadata*: "what else do we know about this coverage?" This item is added in this document, it is not present in the abstract definition of ISO 19123-1:2023.

Technically, all coverage types are derived from abstract class `Coverage`. This structure contains a `DomainSet` describing the coverage's domain and a `RangeSet` component containing the range value consisting of one or more record fields. The `RangeType` element describes the coverage's range data structure. Its structure description is based on the OGC SWE Common [OGC 08-094r1] `DataRecord`, so that the semantics description from upstream sensor acquisitions into downstream services is carried over seamlessly.

In conformance class *coverage*, this domain/range representation is used; requirements class *coverage-partitioning* (Clause 11) adds partitioning and position/value pair list as alternatives. This is why coverage subtype `CoverageByDomainAndRange` is introduced in Figure 3; while it can seem artificial in this requirements class, it will allow modelling the alternative representations later in this document.

Figure 3 contains the complete coverage structure with all variants allowed; in the subsequent subclauses, all parts are successively described in detail.

NOTE ISO 19123-3:2023[42], which is based on the OGC Web Coverage Processing Service (WCPS)[39], defines a coverage-specific query language based on this document allowing extraction, recoding, fusion, derivation of coverages and general analytics, currently on multi-dimensional grid coverages, i.e. geo datacubes.

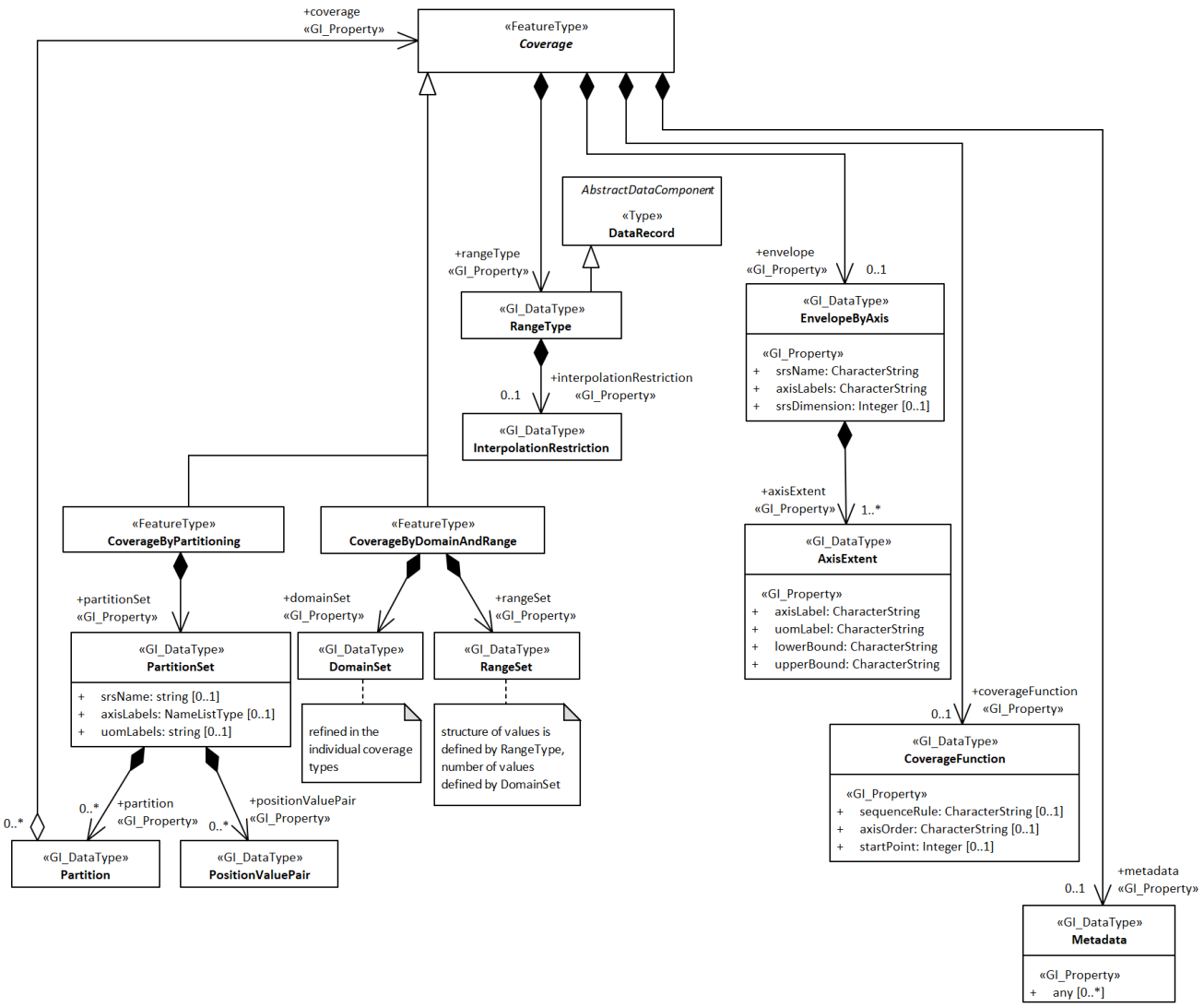

Figure 3 — The `Coverage` structure (overview)

5.3 Domain/range based coverage structure

The coverage structure defined normatively in this class *coverage* is `CoverageByDomainAndRange`.

Requirement 3: https://standards.isotc211.org/19123/-/2/2/req/coverage/domain+range

A coverage instantiating class *coverage* shall conform with Figure 4 and Table 2.

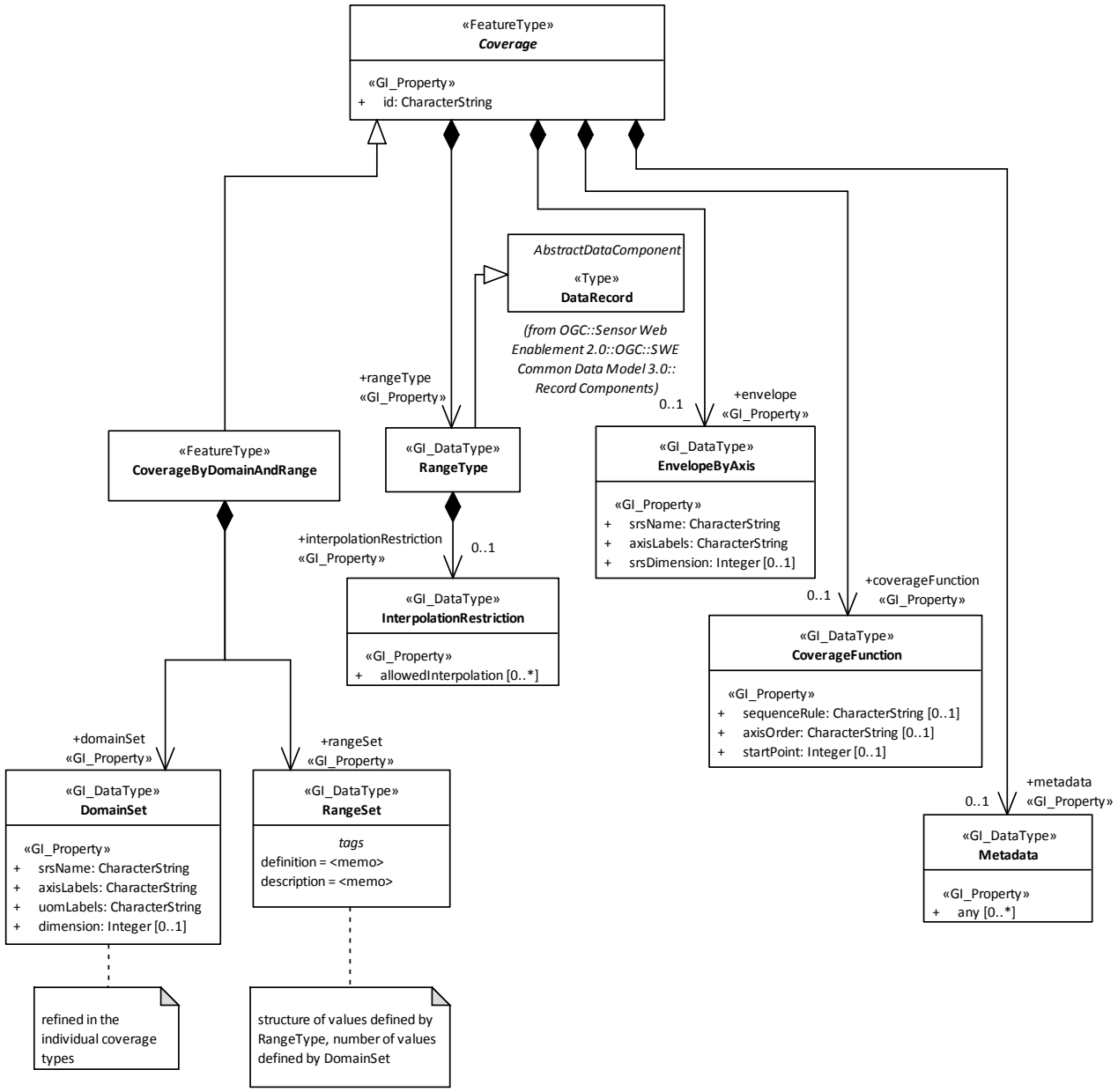

Figure 4 — The `Coverage` domain/range structure

Table 2 — The `Coverage` domain/range structure

Name	Definition	Data type	Multiplicity
`Id`	Identifier of the coverage	`string`	One (mandatory)
`coverage-Function`	Function describing the mapping from the domain to the range of the coverage (multiplicity zero, unless overridden by a coverage subtype)	`Coverage-Function`	Zero or one (optional)
`Envelope`	Bounding box of the coverage	`Envelope-ByAxis`	Zero or one (optional)

Name	Definition	Data type	Multiplicity
domainSet	coverage domain set, specifying the direct positions at which range set values are available in this coverage	DomainSet	One (mandatory)
rangeSet	Coverage range set, containing a value for each direct position in the domain set	RangeSet	One (mandatory)
rangeType	Structure definition of the coverage range values, as specified in OGC 08-094r1 OGC SWE Common 2.0 Clause 7 and 8	SWE Common::DataRecord	One (mandatory)
metadata	Application specific metadata of the coverage	Any	Zero or one (optional)

NOTE 1 In previous editions of this standard, the id attribute was of type NCName which restricts the characters allowed – a legacy from GML. In the course of the separation of logical and physical level and for the support of further formats without this restriction, such as JSON, this has been relaxed. Generally, this results in a more human-readable style allowing for whitespace, special characters, globally unique naming schemes, etc. Care has to be taken, therefore, to choose only id values which can be represented in all target formats envisaged (which can, but does not necessarily include GML).

The same separation from GML is not possible for rangeType as SWE Common does not offer a separate logical model.

NOTE 2 UML data type Any is used here with the same meaning as XML's xsd:any, which does not have a direct equivalent in UML.

The coverageFunction item describes the correlation between the direct positions in the domain and the values in the range. In case of the Multi-Point/Curve/Surface/Solid Coverages this correlation is straightforward: points, curves, surfaces, and solids are listed linearly, and so are the range values, and based on the sequence of occurrence of the items in both lists, pairs can be built unambiguously. Therefore, coverageFunction in these cases is not needed and, hence, has a zero occurrence in general. This is different in gridded coverages as the set of discrete points is aligned in multiple dimensions with no single linearization scheme. Consequently, GeneralGridCoverage provides an explicit sequence definition (cf. Subclause 7.2).

NOTE 3 Although currently coverageFunction applies only to grids in a GeneralGridCoverage the structure remains associated with Coverage as defined in OGC CIS 1.1 for backwards compatibility and to leave open opportunities in future for using it in non-gridded coverage structure, too.

Table 3 — The CoverageFunction structure

Name	Definition	Data type	Multiplicity
sequenceRule	Linearization scheme code, case-insensitive. Default is "Linear"	string	Zero or one (optional)
axisOrder	A list of the grid axes, identified by their decimal position number starting at 1, and ornamented with a "+" for traversal in ascending coordinate order or "-" for traversal in descending coordinate order. If axisOrder is present then each axis shall appear exactly once. Default is "+1 +2 …+*n*" for an *n*-D grid.	string	Zero or one (optional)
startPoin	The *n*-D index position of a point in the *n*-D grid that is mapped to	integer	Zero or one

Name	Definition	Data type	Multiplicity
t	the first point in the range set (the start of the linearization traversal). Default is the *n*-tuple of lower index bounds in the grid.		(optional)

NOTE 4 The axes in axisOrder are identified by their position number, starting with 1. The array start position, given by the lower bounds vector, typically is (0,...,0). This is a GML heritage.

Requirement 4: https://standards.isotc211.org/19123/-/2/2/req/coverage/coverage-function

The coverageFunction item, if present in a Coverage, shall consist of a CoverageFunction structure as per Table 3.

NOTE 5 GML references withdrawn ISO 19123:2005 for the definition of the coverageFunction details. ISO 19123:2005 is superseded by ISO 19123-1:2023 which contains the same information in its (informative) Annex C. As there the traversal variants are given only with a coarse informal description and illustrations of the 2-D case this is not sufficiently well defined for normative use..

5.4 Domain set

5.4.1 General

The domain set determines the exact locations of a coverage overall and its set of direct positions. The coordinate space in which the coverage resides is given by a (single- or multi-dimensional) CRS defining an ordered list of domain set axes whose lower and upper bounds establish the extent along each axis.

A CRS is referenced through some identifier. Axes used by the coverage are identified by their position in the (ordered) list of axes given in the CRS. The srsName attribute contains an identifier which resolves to the complete CRS definition. To avoid this extra roundtrip in services, information critical for understanding the coverage – the axis labels and the unit of measure for each axis – is repeated locally in the domain set as the two lists axisLabels and uomLabels.

Table 4 — The DomainSet structure

Name	Definition	Data type	Multiplicity
srsName	Identifier of the CRS in which the coverage domain set coordinates are expressed	string	One (mandatory)
axisLabels	List of whitespace-separated pairwise distinct axis names, each one corresponding to exactly one axis in the CRS, matching the position in the axis name list and the axis position in the CRS. Axis names do not contain whitespace. They can be identical to the respective CRS axis abbreviation, but do not have to.	string	One (mandatory)
uomLabels	List of whitespace-separated units of measure (uom), where each uom belongs to the axis matched by the position in axisLabels. Uom items do not contain whitespaces.	string	One (mandatory)
dimension	Dimension of the coverage, given by its CRS	unsigned int	Zero or one (optional)

NOTE 6 Attribute dimension is redundant (the dimension is equal to the number of elements in the axisLabels and uomLabels lists) and hence not present in Multi-Point and General Grid Coverage domain sets, only in the legacy GML Multi-Curve/Surface/Solid coverage domain sets.

EXAMPLE Examples of unit labels include “deg” (degree), “m” (metre), and “1” (for unit-less scalars).

Requirement 5: https://standards.isotc211.org/19123/-/2/2/req/coverage/srsname-contents

The `srsName` attribute shall reference a CRS containing all axes referenced in the domain set in proper order.

Requirement 6: https://standards.isotc211.org/19123/-/2/2/req/coverage/axislabels-contents

The `axisLabels` attribute shall consist of a whitespace-separated list of names, with exactly as many names as the `srsName` CRS defines.

Requirement 7: https://standards.isotc211.org/19123/-/2/2/req/coverage/uomlabels-contents

The `uomLabels` attribute of an axis shall consist of a whitespace-separated list of units of measure (uom) items, with exactly as many uom items as the `srsName` CRS defines.

Requirement 8: https://standards.isotc211.org/19123/-/2/2/req/coverage/srsname-crs

The CRS in the `srsName` attribute shall have as many axes as indicated in the `dimension` attribute, if that is present.

NOTE 2 See Subclause 5.4.2 for details on the CRS specification.

NOTE 3 In the context of coverage services like WCS the domain set CRS is called the coverage’s Native CRS, as opposed to derivatives of this coverage in some other CRS, obtained through reprojection.

NOTE 4 Axis labels can be renamed locally in the coverage. For example, CRS axes Lat / Lon can be named Lat / Long or x / y. Matching is done solely by their position.

NOTE 4 As a consequence of these requirements, `axisLabels` and `uomLabels` contain the same number of elements, equal to the coverage’s dimension.

NOTE 5 UCUM and QUDT are common methods for expressing units of measure.

5.4.2 Coordinate reference system

Each coverage has a single coordinate reference system (CRS) associated which defines the meaning of the direct position coordinates across all axes of the coverage. This CRS has the same dimension as the coverage domain set. The description of this CRS can be given by a single predefined CRS (such as from the EPSG catalogue) or a composition of several CRSs (such as EPSG horizontal and OGC time).

By convention established by OGC, a reference to a CRS is a URL pointing to a resource which is a CRS definition. As URLs for humans are impractical, a best practice has been established by OGC to alternatively allow a shorthand notation using SafeCURIE syntax[40]. This syntax is defined as follows:

— The CRS URLs to be abbreviated shall follow the pattern (with placeholders in curly braces) https://www.opengis.net/def/crs/{authority}/{version}/{identifier}

— Then, a valid CURIE equivalent for such a CRS URL is `[{authority}:{identifier}]` with `{version}` in the CURIE set to 0 by definition, always taking the most up-to date version.

— For composite CRSs using the *crs-compound* pattern to chain several CRSs, the CURIE equivalent is their concatenated comma-separated list `[{authority`$_1$`}{identifier`$_1$`}], ..., [{authority`$_n$`}{identifier`$_n$`}]`

EXAMPLE The following are valid CRS URLs and corresponding CURIEs, based on the OGC resolver operated by Constructor University (line breaks added to improve readability):

https://www.opengis.net/def/crs/EPSG/8.5/4326 [EPSG:4326]

https://www.opengis.net/def/crs-compound? [EPSG:4326],[OGC:AnsiDate]

1=https://www.opengis.net/def/crs/EPSG/0/4326&

2=https://www.opengis.net/def/crs/OGC/0/AnsiDate

NOTE In the previous version of this standard (and likewise in Annex B, for maintaining compatibility) the identifiers referenced in `axisLabels` had to be identical to the corresponding `axisAbbrev` value in the CRS definition. In this version, coverage `axisLabels` and CRS `axisAbbrev` are decoupled so that there is no longer such dependency. This definition is backwards compatible, i.e. coverages can continue to use CRS `axisAbbrev` values; however, `axisAbbrev` values in subsequent versions of a CRS (such as from EPSG) mcan potentially change without notice, so the correspondence can get lost over time.

5.4.3 Direct positions

Coverages contain a finite number of positions for which data are stored, called "direct position". Depending on the coverage type this can be points, curves, surfaces or solids. Depending on the interpolation available (see Subclause 0) further values can be derived between these direct positions.

Coordinates are of data type string in this document as they are required to accommodate data types as diverse as numbers (such as 1.23 degrees or 500 nm), dates and times (such as "2016-03-08T11:23Z"), categorial values (such as "orange", "apple"), and possibly more. Similarly, resolution specifications are of type string as they have to accommodate, e.g., "1.23" for degrees or metres and "PT2h" for a 2-hour duration. As per ISO 19111:2019, any coordinate representation scheme is required to convey some total ordering so that expressions like "lowerBound ≤ upperBound" are valid for any axis.

NOTE 1 This extension deviates from traditional GML which allows numbers only.

A direct position is described through n-dimensional coordinate vectors. The order of the coordinates in a direct position vector is given by the axis order of the coverage's Coordinate Reference System (CRS), also available locally in the coverage through the `srsName` attribute. Any combination of spatial, temporal, and "abstract" (i.e. non-spatio/temporal) axes is possible. The basis for representing direct positions of coverages is the type `DirectPosition`.

For points this Coverage Implementation Schema defines such a generic type, `DirectPositionPoint`. The higher-dimensional geometric elements – curves, surfaces, and solids – use GML to not invent parallel structures.

NOTE 2 This implies that, due to the GML legacy, for curves, surfaces, and solids currently only numerical coordinates are currently available.

Requirement 9: https://standards.isotc211.org/19123/-/2/2/req/coverage/directposition

In a `GeneralGridCoverage` and `MultiPointCoverage` the coordinates in direct positions shall be a `DirectPositionPoint` as described by Figure 5.

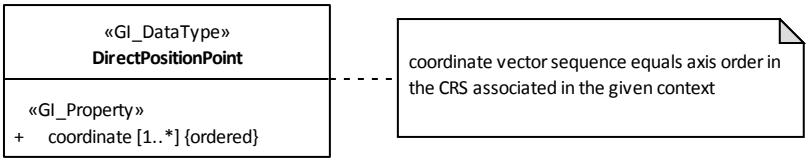

Figure 5 — `DirectPositionPoint` structure

5.4.4 Envelope

The optional `Envelope` component helps applications in gaining a quick overview of the coverage's location, in case the coverage coordinates are expressed in some more uncommon CRS.

NOTE 1 Particularly in the presence of displaced axes, transformation axes, and discrete coverages the domain set can quickly become hard to oversee.

The envelope does not need to use the same CRS as the domain set, therefore the bounding box will not necessarily be the minimal. However, it always completely encloses the coverage.

NOTE 2 As with the domain set, URIs and CURIEs can be used for identifying CRSs.

The structure of an envelope is a list of the coverage axes and the corresponding extents along each axis, expressed by their lower and upper bounds. Additionally, the CRS, the axis order and the corresponding units of measure along the axis are provided.

Requirement 10: https://standards.isotc211.org/19123/-/2/2/req/coverage/envelopebyaxis

If present, the envelope of a coverage instantiating class *coverage* shall consist of an `EnvelopeByAxis` element conforming to Figure 6, Table 4, and Table 6.

Requirement 11: https://standards.isotc211.org/19123/-/2/2/req/coverage/envelope-srsname

In a coverage `EnvelopeByAxis` the `srsName` attribute shall represent a CRS which has the same dimension as the CRS in the `domainSet` attribute `srsName`.

Requirement 12: https://standards.isotc211.org/19123/-/2/2/req/coverage/envelope-axislabels

In a coverage `EnvelopeByAxis` the `axisLabels` attribute shall consist of a whitespace-separated list of pairwise distinct names, with exactly as many names as defined by the envelope's `srsName` CRS.

Requirement 13: https://standards.isotc211.org/19123/-/2/2/req/coverage/envelope-axisextent

In a coverage `EnvelopeByAxis` there shall be one `axisExtent` item for each name in the `axisLabels` attribute with identical name in `axisExtent.axisLabel`.

Requirement 14: https://standards.isotc211.org/19123/-/2/2/req/coverage/envelope-uomlabel

In a coverage `EnvelopeByAxis`, in each `axisExtent` the `uomLabel` value shall be the unit of measure as defined in the CRS for this axis.

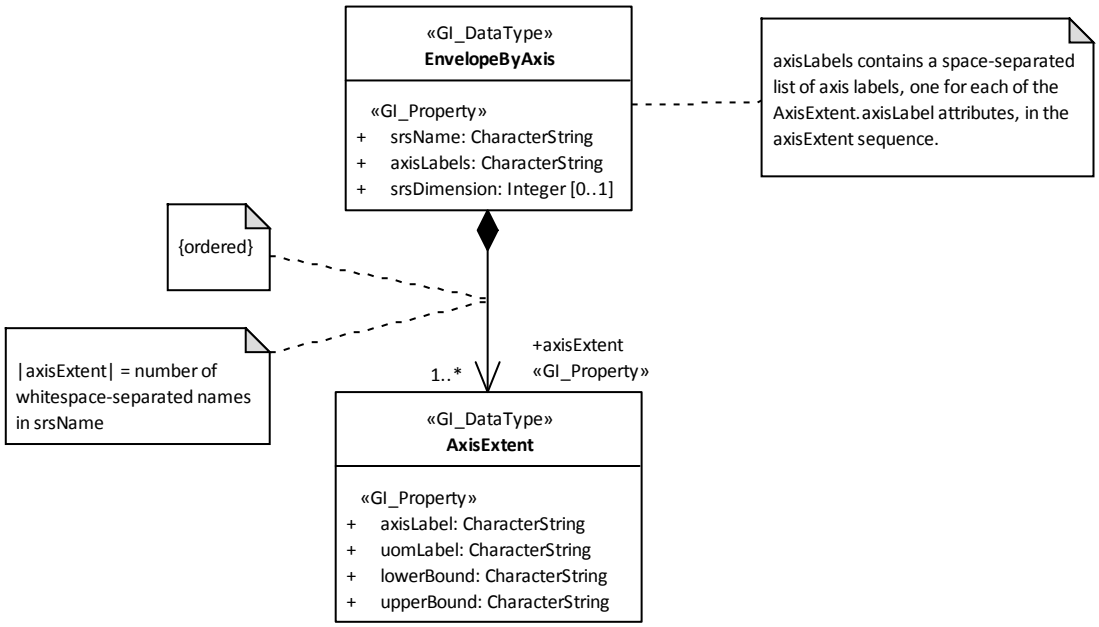

Figure 6 — `EnvelopeByAxis` structure

Table 5 — `EnvelopeByAxis` structure

Name	Definition	Data type	Multiplicity
`srsName`	Identifier of the CRS in which axis extent coordinates are expressed	`string`	One (mandatory)
`axisLabels`	List of all axis names appearing in `axisExtent`, space-separated, in order, each exactly once	`string`	One (mandatory)
`axisExtent`	Bounding interval along a specific axis	`AxisExtent`	One or more (mandatory)
`srsDimension`	CRS dimension (i.e., number of items in `axisLabels`) Note: this is redundant and therefore optional	`unsigned integer`	Zero or one (optional)

Table 6 — `AxisExtent` structure

Name	Definition	Data type	Multiplicity
`axisLabel`	Axis identifier	`string`	One (mandatory)
`uomLabel`	Shorthand identifier of the Unit of Measure used on this axis (as indicated in the CRS definition for this axis)	`string`	One (mandatory)
`lowerBound`	Lowest coordinate along this axis	`string`	One (mandatory)

upperBound	Highest coordinate along this axis	string	One (mandatory)

NOTE 3 It is recommended as a Best Practice to use UCUM[30] for the units of measure.

Requirement 15: https://standards.isotc211.org/19123/-/2/2/req/coverage/envelope-lowerbound-upperbound

For each axisExtent in the EnvelopeByAxis element of a coverage the lowerBound shall be less than or equal to the upperBound.

Requirement 16: https://standards.isotc211.org/19123/-/2/2/req/coverage/envelope-domainset

In a coverage Envelope (if present), all direct positions shall lie completely inside the DomainSet along all coverage dimensions.

5.5 Range type

5.5.1 Overview

The RangeType component adds a structure description and technical metadata required for understanding the coverage range values. As such, it defines, to some extent, the range set semantics. For this structure description, the SWE Common DataRecord is used. Optionally, interpolation directives can be added.

NOTE By relying on the data description of SWE the semantics of sensor data can be carried over into coverages without loss of information.

Requirement 17: https://standards.isotc211.org/19123/-/2/2/req/coverage/rangetype

The RangeType of a coverage shall have a structure as given in Table 7.

Table 7 — RangeType structure

Name	Definition	Data type	Multiplicity
DataRecord	Description of the common data type of all range values	SWE Common:: DataRecord (OGC 08-094r1)	One (mandatory)
Interpolation Restriction	Interpolation methods meaningfully applicable to this coverage	InterpolationRestriction	Zero or one (optional)

5.5.2 Data description

Specification of the common data type all range values share is done through the DataRecord part of the coverage's RangeType component. The structuring mechanisms used with range values are records.

Requirement 18: https://standards.isotc211.org/19123/-/2/2/req/coverage/datarecord

The range type component of a coverage shall conform with the DataRecord of SWE Common [OGC 08-094r1].

Requirement 19: https://standards.isotc211.org/19123/-/2/2/req/coverage/record-or-array

Wherever SWE Common [OGC 08-094r1] allows an `AbstractDataComponent` in a coverage range structure the concrete instance shall be one of the subtypes `DataRecord` and `DataArray`.

Within a `DataRecord` contained in a concrete range structure, each of its record components is locally uniquely identified by the record component's `field` attribute, in accordance with the "soft-typing" property introduced by SWE Common.

Atomic data types available for range values are those given by the SWE Common data type `AbstractSimpleComponent`. As a range structure contains only structure definitions, but not the values themselves (these sit in the coverage range set component), the optional `AbstractSimple-Component` component `value` is suppressed in coverages, likewise the `encoding` component.

Requirement 20: https://standards.isotc211.org/19123/-/2/2/req/coverage/no-value-component

For all `SWE Common` [OGC 08-094r1] `AbstractSimpleComponent` items in the range type of a coverage, instance multiplicity of the `value` component shall be zero.

Due to the range type reliance on SWE Common [OGC 08-094r1], its use in coverages to a large extent is defined there. For the reader's convenience, key parts of the `field` in a `DataRecord` are listed non-normatively below:

— `name` (mandatory): an identifier, unique within a record, used for accessing the record component in "range subsetting" like in WCS and WCPS. Example: "red".

— `definition` (mandatory): a URI that resolves to the human readable definition of the property. Example: https://ec.europa.eu/eurostat/web/gisco/geodata/population-distribution/population-grids .

— `nilValues` (optional): a list of values / value intervals which are to be interpreted as placeholders for values not present (for example, to be ignored in WCPS aggregation queries). Each list entry consists of the value / interval together with a reason URL pointing to the definition and meaning of this nil. Example: (-INF, http://www.opengis.net/def/nil/OGC/0/BelowDetectionRange".

— `uom` (unit of measure, optional): SWE Common mandates UCUM[43], a human comprehensible, yet compact and unambiguous syntax[1]. Units are used with `SWE:Quantity` only, not for `SWE:Category` nor `SWE:Count`. Examples: "m/s" for speed, "W.m-2.Sr-1.um-1" (watts per square meter per steradian per micrometer) for radiance:

Further, SWE Common distinguishes coverage-relevant data categories `SWE:Quantity`, `SWE:Count`, and `SWE:Category`. `Quantity` models data with a continuous numerical expressed in the unit of measure indicated. `Count` models discrete, countable data, hence is unitless. `Category` models characteristics, often represented by numbers referring to a dictionary establishing the meaning; values are unitless. Distinguishing these types is important to govern summarizability – for example, categorical data, such as land classification, allow only nearest-neighbor interpolation in scaling whereas optical satellite imagery allows further types of interpolation.

[1] QUDT was suggested at some time as an alternative, as it is part of semantic definitions allowing reasoning. However, QUDT results in long URLs not useful for human consumption – a disadvantage which for CRSs has led to accept CURIEs in addition to URLs.

5.5.3 Interpolation

Via interpolation range values can be obtained for positions in a coverage which are not identical to some direct position, if one or more interpolation methods are mentioned in the coverage. Coverages where values are available only for direct positions are called “discrete” whereas coverages where interpolation allows deriving in-between values are called “continuous”.

EXAMPLE A satellite image can be interpolated by *nearest neighbour*, *linear*, *quadratic*, and several more methods. A land classification map, on the other hand, can only be interpolated using *nearest-neighbour*.

The optional `InterpolationRestriction` element governs interpolation of the coverage:

— Without an `InterpolationRestriction` element, any interpolation is admissible on the coverage (for backward compatibility).

— Inside an `InterpolationRestriction`, each `allowedInterpolation` indicates an interpolation method which can be meaningfully applied to the coverage on hand (hence, it is allowed).

— An empty `InterpolationRestriction` list means that no interpolation is applicable at all.

NOTE 1 As selection of a particular interpolation method is in the hands of the application which processes a coverage, this is not a testable behaviour of the coverage data structure and, therefore, it cannot be put into a formal, testable requirement. Instead, a corresponding test needs to be added to any coverage service definition which includes interpolation (such as the WCS Interpolation extension).

Table 8 — `InterpolationRestriction` structure

Name	Definition	Data type	Multiplicity
`allowedInterpolation`	interpolation method which can be meaningfully applied to this coverage	`anyURI`	Zero or more (optional)

Requirement 21: https://standards.isotc211.org/19123/-/2/2/req/coverage/interpolation

An **`InterpolationRestriction`** element, if present, shall conform with Table 8.

NOTE 2 Strictly speaking, every interpolation method defines a separate coverage with different (interpolated) range values. For efficiency reasons, however, it makes sense to combine otherwise identical coverages differing only in the interpolation methods into one coverage.

5.6 Range set

The range set contains the actual values, each of which is associated with one direct position as defined in the domain set. All range values share the data type given by the range type.

It is necessary to have a 1:1 correspondence between direct positions in the domain set and range values in the range set. Neither duplicates nor values omitted are allowed.

NOTE 1 For range values not known, null values can be used for “gap filling” if defined in the range type.

Requirement 22: https://standards.isotc211.org/19123/-/2/2/req/coverage/one-value-per-grid-position

For each direct position in the `domainSet` of a coverage (excluding `gridLimits` if existing) there shall exist at least one (atomic or composite) value in the coverage’s range set, based on the structure shown in Figure 7.

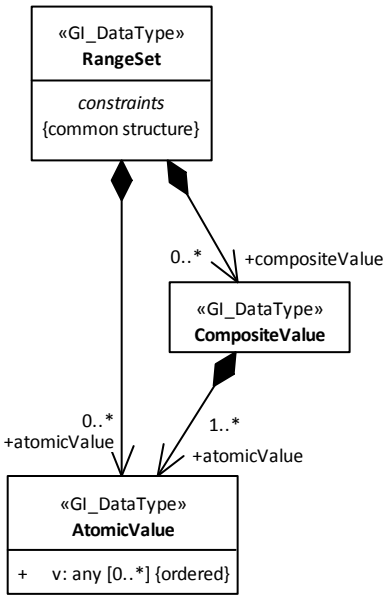

Figure 7 — `RangeSet` structure

NOTE 2 Direct positions can repeat coordinates (as it happens, for example, with point clouds), and each occurrence can have its individual range value. Requirement 22, therefore, talks about "positions listed in the domain set" and not about "coordinate positions" as such. Only in the grid coverages specified in Clause 7 is only one value per direct position coordinate allowed (see Requirement 26).

The data type of all range values is the same, it is given by the range type defined through a `SWE::DataRecord`. In particular, atomic values inside a composite value shall be listed exactly in the same sequence as the range type components.

Atomic values inside a composite value shall be listed exactly in the same sequence as the range type components prescribe.

Requirement 23: https://standards.isotc211.org/19123/-/2/2/req/coverage/rangetype-adherence

All values in the `rangeSet` of a coverage shall be consistent with the structure description provided in its `rangeType`.

5.7 Metadata

The optional `metadata` component is a carrier for any kind of application dependent metadata. Hence, no requirements are imposed here.

NOTE Implementations can impose restrictions on metadata stored (such as on their sheer volume or some particular XML / JSON schema to be used).

6 Multi-Point Coverage

This clause establishes conformance class "multipoint".

A Multi-Point Coverage (also called "point cloud") is an unordered collection of arbitrarily distributed geometric points. Technically, a `MultiPointCoverage` contains a domain set of type `MultiPoint`.

Requirement 24: https://standards.isotc211.org/19123/-/2/2/req/pointcloud/dependency

A coverage instantiating class multipoint shall conform with class *coverage*.

Requirement 25: https://standards.isotc211.org/19123/-/2/2/req/pointcloud/domainset

In a `MultiPointCoverage`, the `DomainSet` **shall** consist of a `MultiPoint`.

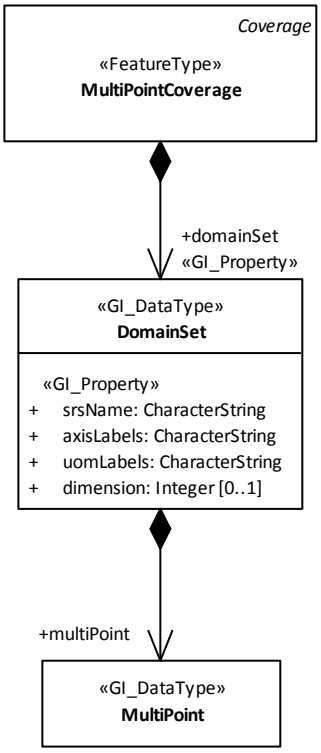

Figure 8 — UML diagram of `MultiPointCoverage` structure

NOTE MultiPoint has been redefined in CIS over its original definition in OGC CIS 1.0 to allow timeseries of multi-points, aka point trajectories.

7 General Grid Coverage

7.1 Overview

This Clause establishes conformance classes *grid-regular*, *grid-irregular*, and *grid-transformation* as special cases of the general grid coverage.

7.2 General grid

A grid coverage serves to model raster data and datacubes. Logically, it is a special case Multi-Point Coverage where the direct positions are aligned on some multi-dimensional grid. A grid is an ordering scheme for points where every direct position along every axis has exactly one nearest neighbour position in positive and one in negative axis directions (cf. Figure 9), except for the grid boundaries where outside the grid domain no such neighbours exist. The minimum distances between two direct positions can be identical along the grid extent (“regular grid”), or they can vary (“irregular grid”).

A specialty of the gridded coverages is that for each direct position there is exactly one associated range value (not more than one as is possible with the other coverage types).

All these specific properties together lead to separate classes for grid coverages, independent from class *multipoint*.

Requirement 26: https://standards.isotc211.org/19123/-/2/2/req/grid-regular/exactly-one-value-per-grid-position

For each direct position in the `domainSet` of a `GeneralGridCoverage` there shall exist exactly one (atomic or composite) value in the range set.

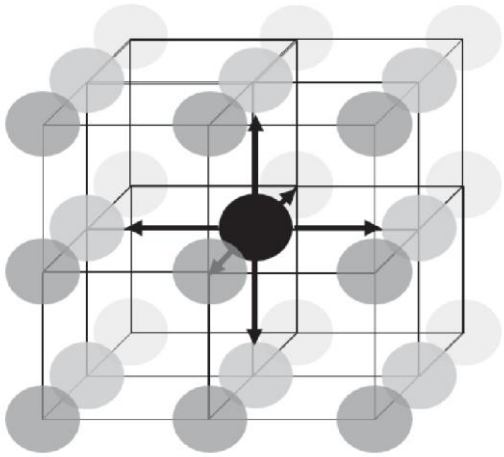

Figure 9 — Multi-dimensional neighbourhood in a grid

The coverage grid is described in the coverage's domain set through a single multi-dimensional CRS establishing a non-empty list of coordinate axes and their properties. These axes span the grid, thus defining the coordinate tuples of the direct positions the grid offers. To this end, the (otherwise unchanged) coverage structure has in its `DomainSet` a `GeneralGrid` with various axis types: index (non-georeferenced, with integer coordinates and no unit of measure), regular georeferenced (with space, time, or other coordinates), irregular, warped, and transformation. In a given coverage, each axis has its individual axis type. The following subclauses establish these axis types and corresponding conformance clauses in turn.

NOTE 1 The special case that all axes are regular corresponds to a Rectified Grid, whereas a grid with at least one irregular axis corresponds to a Referenceable Grid. These legacy grid coverage definitions are included in normative Annex B.

NOTE 2 Skewed and rotated grids are not modelled explicitly in CIS; they can be represented by making the grid's CRS a concatenation of any given CRS with an Engineering CRS describing, e.g., any affine transformation of the original grid, in accordance with ISO 19111:2019.

Underlying the n-D grid is an n-D array holding the range values of the direct positions. As is common in programming, this array is addressed in integer coordinates, equivalent to index axes described through an index CRS. As many algorithms just iterate over this bare array, ignoring the domain coordinates, the coverage provides information on how to access the range values along the grid on index level. Two components provide the necessary information. The `gridLimits` item contains, for each dimension in turn, the lower and upper index array bound. The `coverageFunction` describes how the multidimensional grid index coordinates map to the linearized representation in the range set.

NOTE 3 For the clarification of nomenclature, a referenced coverage has two grids (typically, spatio-temporal), a referenced one defined through the coverage's native CRS, and an integer-based array grid (trivially, with an Index

CRS, but that is omitted). A non-referenced coverage (such as a mathematical matrix) has only one grid, the array with its unit-less coordinates.

If a coverage's grid consists exclusively of index-type axes then obviously the grid limits indication is redundant, and therefore can be left out. In this case, the coverage function refers directly to the domain axes, rather than an array specified by the grid limits.

Earlier versions of this standard adopted the grid coverage classification based on the nature of the entire grid of a coverage, distinguishing Rectified Grid Coverage and Referenceable Grid Coverage. This is superseded by the unifying concept of a General Grid Coverage which allows a more fine-grain distinction based on the nature of each coverage domain axis. Rectified and Referenceable Grid Coverage resemble special cases in this perception:

— A Rectified Grid Coverage is a grid coverage where every axis is either an index axis or a regular axis;

— A Referenceable Grid Coverage is a grid coverage where at least one axis is neither an index axis nor a regular axis.

Both Rectified and Referenceable Grid Coverage structures, therefore, can be modelled through General Grid Coverage. For backwards compatibility, Rectified and Referenceable Grid Coverages are still retained in this standard, but – given their slightly different internal structure – form a separate conformance class rectified+referenceablegridcoverage which shall be as specified in Annex B/which shall conform to the specifications provided in Annex B. Recommendation to implementers of coverages is to use `GeneralGridCoverage` as defined in this Clause 7 and to phase out the legacy coverages as defined in Annex B.

Given the multi-dimensional alignment of the direct positions in a grid it is necessary to define the correspondence between grid positions and the linear range sequence. This is achieved through the `CoverageFunction` component which resembles a grid mapping function acting as linearization of the multi-dimensional domain set, specifically: the grid. Therefore, all grid coordinates under discussion are integer, referring to the grid limits (cf. Table 10). The mapping consists of three parts (cf. Table 3):

— The linearization scheme in `sequenceRule`. Currently, only `Linear` is supported normatively in this document. Implementations may support other schemes, but should document them thoroughly.

  In linear scanning, range values are assigned to consecutive grid points along a single grid line parallel to the first grid axis listed in scanDirection. Once scanning of that row is complete, assignment of feature attribute value records steps to another grid line parallel to the first and continues to step from grid line to grid line in a direction parallel to the second axis. If the grid is 3-dimensional, the sequencing process completes the assignment of feature attribute value records to all grid points in one plane, then steps to another plane, then continues stepping from plane to plane in a direction parallel to the third axis of the grid. The process can be extended to any number of axes. Linear scanning is continuous only along a single grid line.

— In `axisOrder`, the linearization parameters determining the details of linearization. In case of `Linear`, this consists of an indication of the coverage's axes (identified by their decimal position number in the grid) together with the direction of traversal for each axis. This direction is indicated by a + or - sign prefix.

— In `startPosition`, the starting point for the linearization traversal.

If the `coverageFunction` item is omitted for a gridded coverage, `sequenceRule` is assumed to be `Linear`, `axisOrder` is assumed to be "+1 +2 …" (referencing all grid dimensions), and `startPoint` is assumed to be the lowest grid coordinate,

Requirement 27: https://standards.isotc211.org/19123/-/2/2/req/coverage/grid-coverage-function

In a `GeneralGridCoverage`, the `coverageFunction` item shall have a multiplicity of zero or one, with the following component value rules:

— `sequenceRule` shall have a value of "Linear" (case insensitive), which also is the default if `coverageFunction` is omitted entirely.

— `axisOrder` shall encode a whitespace-separated list of the axes making up the grid, or (if all axes in the coverage's CRS are of index-type), the list of (index) axes making up the coverage's domain, as defined in its CRS. An optional sign prefix indicates positive axis direction with "+" and negative axis direction with "-". Default if no `coverageFunction` is present: "+1 +2 +3 …"

— `startPoint` shall contain the index coordinate of the start of the linearization traversal, as an *n*-tuple for an *n*-D grid. Default if no `coverageFunction` is present is the coordinate tuple of lowest grid point.

Example "+1 -2 +3" means that the points are to be traversed from lowest to highest on the 1st axis, starting at the highest value on the 2nd axis and the lowest value on the 3rd axis points, incremented fastest on the 1st axis before incrementing on the 2nd axis and finally the 3rd.

In the sample range (*i,j*) array shown below, with grid limits (0,4) in *i* and (0,2) in *j* direction, with the default sequence rule "+1 +2", this matrix corresponds to the linearized range sequence 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15, with position numbering from 0 through 14:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

NOTE When programming an iteration over the range values representing the linearized array of the grid, the coordinates of the first axis mentioned in `axisOrder` will change fastest and the coordinates of the last axis mentioned will change slowest when iterating in a nested loop. Each loop runs from the lowest to the highest grid coordinate along its axis for a positive traversal indicator, and downwards otherwise.

EXAMPLE Figure 10 through Figure 13 illustrate four linear 2-D variants, assuming grid index axis order is x (drawn horizontally) first and y (drawn vertically) second.

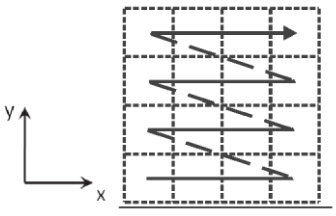

Figure 10 — Schematic linear scanning in a 2-dimensional grid in (x,y) order, sequence "-2 +1"

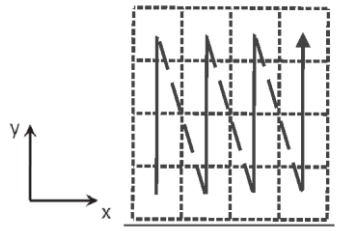

Figure 11 — Schematic linear scanning in a 2-dimensional grid in (x,y) order, sequence "+1 +2"

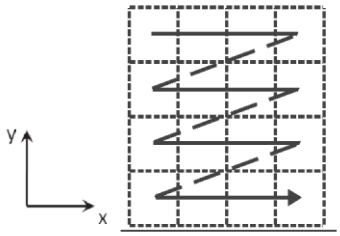

Figure 12 — Schematic linear scanning in a 2-dimensional grid in (x,y) order, sequence "-2 +1"

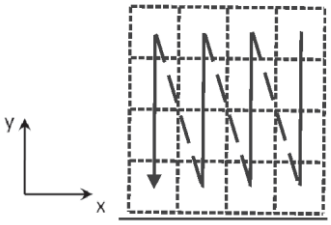

Figure 13 — Schematic linear scanning in a 2-dimensional grid in (x,y) order, sequence "-1 -2"

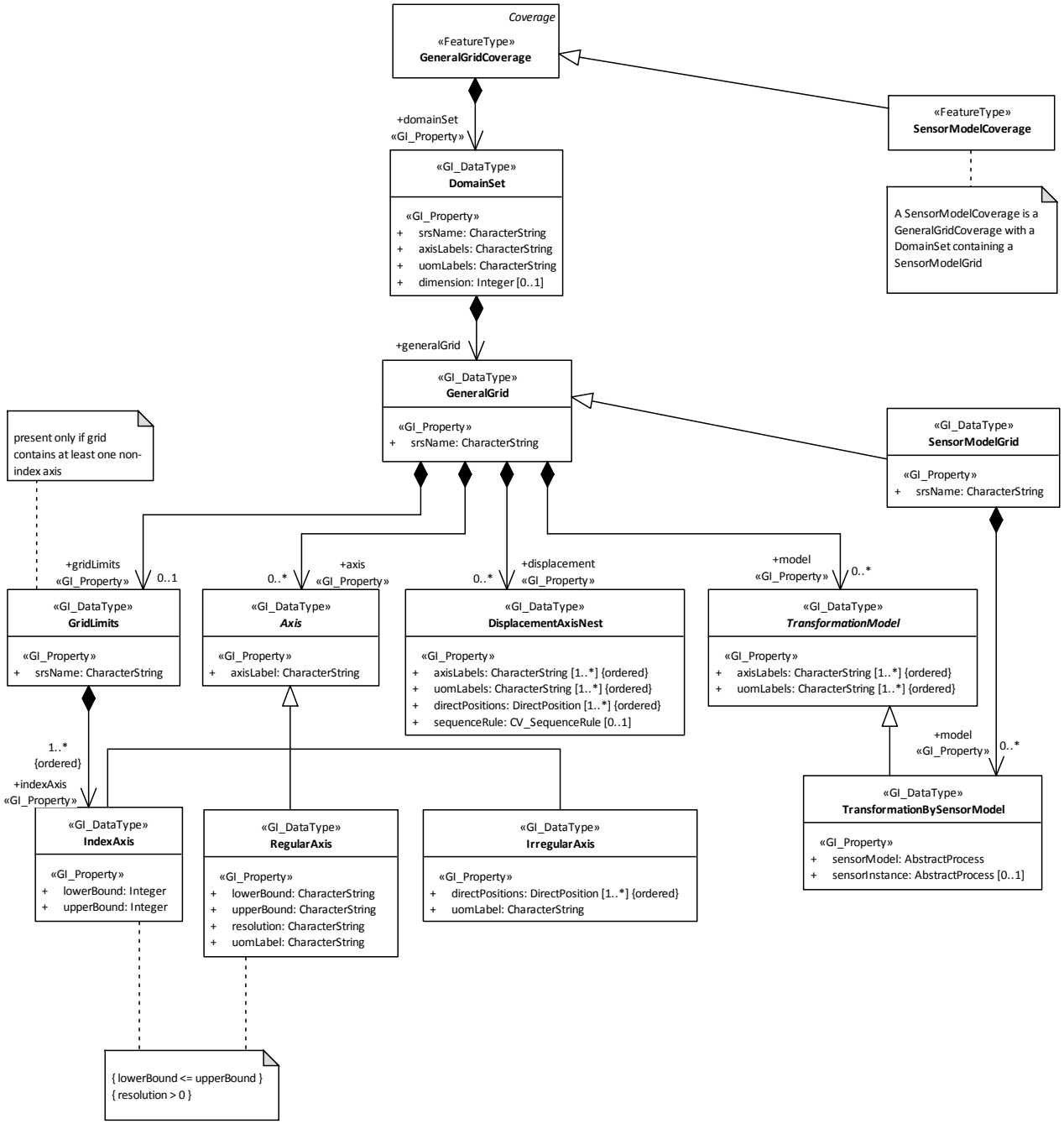

Figure 14— `GeneralGrid` structure (overview)

In the class descriptions of the following subclauses each part of the diagram in Figure 14 is defined in detail.

NOTE A General Grid does not contain global offset vectors because these are available with the axis subtypes where appropriate. It does not contain a rotation vector as this can be modelled by concatenating the CRS with an appropriate Engineering CRS for general affine transformations. Further, skewed and rotated grids are not modelled explicitly in CIS; they can be represented by making the grid's CRS a concatenation of any given CRS with an Engineering CRS describing, e.g., any affine transformation of the original grid, in accordance with ISO 19111:2019.

7.3 Regular grid axis

7.3.1 Overview

Class “grid-regular” establishes coverages with regular (equi-distantly spaced) grid types, both georeferenced and non-georeferenced.

Requirement 28: https://standards.isotc211.org/19123/-/2/2/req/grid-regular/dependency

A coverage instantiating class grid-regular shall conform with class *coverage*.

Requirement 29: https://standards.isotc211.org/19123/-/2/2/req/grid-regular/structure

A `GeneralGridCoverage` shall have a structure as given by Figure 15, Table 9, Table 10, Table 11, Table 12, and Table 13.

Requirement 30: https://standards.isotc211.org/19123/-/2/2/req/grid-regular/domainset

The `DomainSet` of a `GeneralGridCoverage` shall consist of a `GeneralGrid`.

All combinations of axis types index and regular (from class grid-regular), irregular and displaced (from dependent class grid-irregular) are permitted in a coverage domain set. However, no two axes may have the same name (i.e., axis label).

Requirement 31: https://standards.isotc211.org/19123/-/2/2/req/grid-regular/distinct-axis-names

In the `domainSet` of a coverage all axis names shall be pairwise distinct.

EXAMPLE In a Lat/Long/t timeseries datacube, axes Lat and Long describe the geodetic coordinates of the range values and axis t the image acquisition timestamps.

Index axes have integer coordinate values, with a stepping (“resolution”) of 1 and no unit of measure (indicated as “1” in the uom slot where required) and positive axis direction. They are used to create Cartesian grids. The corresponding CRS is an Index CRS.

NOTE Index CRS definitions are provided by the OGC CRS resolver in several dimensions as Index1D, Index2D, etc. with URLs like https://www.opengis.net/def/crs/OGC/0/Index1D and shorthand notation [OGC:Index1D].

Requirement 32: https://standards.isotc211.org/19123/-/2/2/req/grid-regular/index+regular-axes

In a `GeneralGrid`, index (Cartesian) and regular axes shall conform with Figure 15, Table 11, and Table 10.

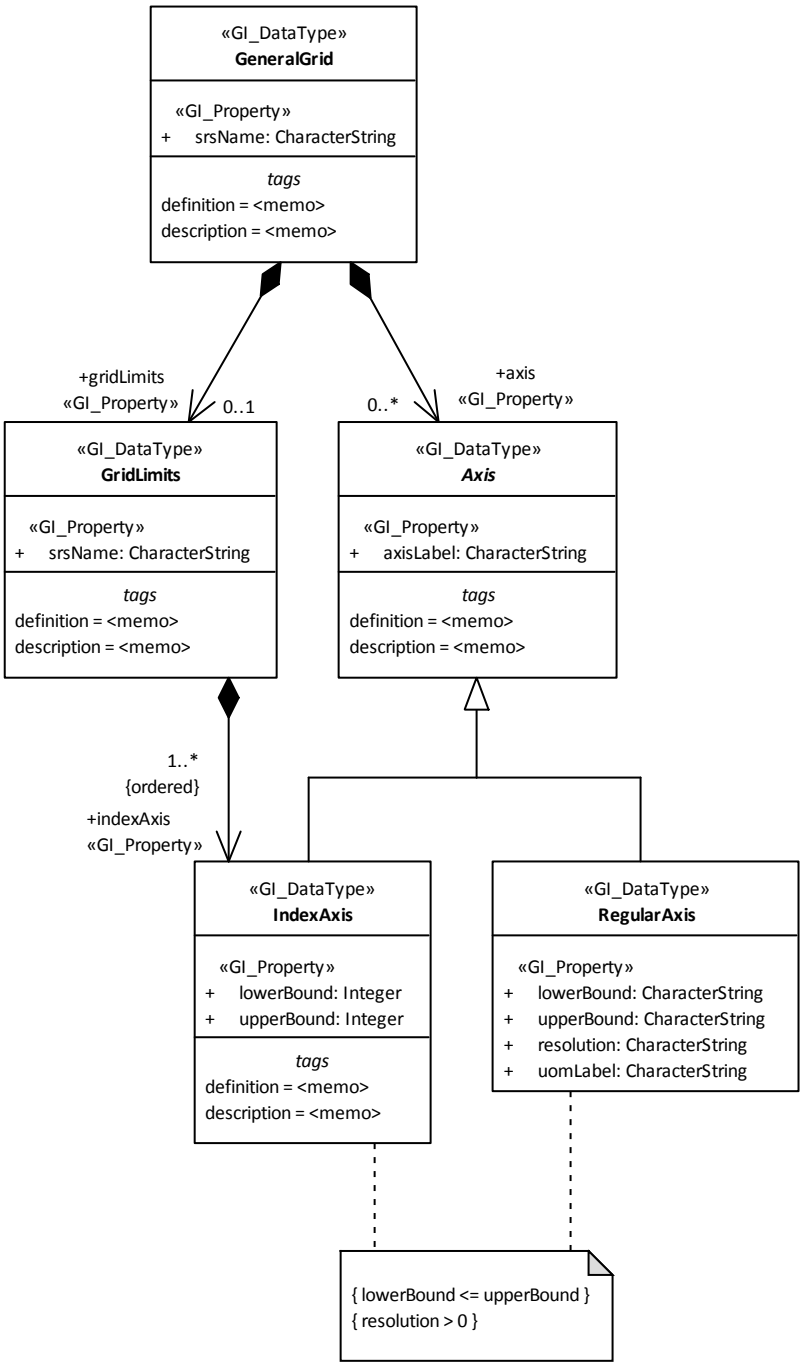

Figure 15 — `GeneralGrid` structure as per grid-regular

Table 9 — `GeneralGrid` structure as per grid-regular

Name	Definition	Data type	Multiplicity
`gridLimits`	grid extent specification, including the (index) CRS in which `axis` coordinates are expressed. If omitted, all lower index boundaries of the axes are assumed 0 (zero).	`GridLimits`	Zero or one (optional)

axis	Axis boundaries, one for each of the axes defined in srsName, in that order	Axis	One to many (mandatory)

GridLimits is an auxiliary structure describing the bare array underlying the grid, hence it is expressed in Cartesian coordinates, using the same mechanism as index axes. This structure is optional because it is not needed when all coverage axes are of type indexAxis, in which case the boundary information is redundant.

Table 10 — GridLimits structure

Name	Definition	Data type	Multiplicity
srsName	Index CRS of the domain set grid array in this coverage	string	One (mandatory)
indexAxis	all axes of the Index CRS referenced in srsName, in proper sequence	IndexAxis	One or more (mandatory)

An Axis item contains information about a particular axis: its axis name. Further information is added in the subclasses, IndexAxis and RegularAxis. Only axes listed in the axisLabels string are allowed, but some axisLabels items may not be present in case further axis types (beyond index and regular) appear in the coverage.

Table 11 — Axis structure

Name	Definition	Data type	Multiplicity
axisLabel	identifier of this axis	String	One (mandatory)

EXAMPLE The Index CRS for a 2-D grid is https://www.opengis.net/def/crs/OGC/0/Index2D, which can also be written as [OGC:Index2D]. It defines axis names *i* and *j*.

Domain sets are required to have a 1:1 correlation between the axis names given in axisLabels and gridLabels, i.e.: they are required to relate pairwise, given by their sequence position. For example, in GeneralGrid attribute settings axisLabels="Lat Lon h date" and GridLimits axisLabels="i j k l" imply a correspondence of Lat with i, Lon with j, h with k, and date with l. On coverage instance level, though, this cannot be conformance tested, therefore this is not a formal requirement.

7.3.2 Index Axis

Axis type IndexAxis requires an Index CRS as its CRS, as defined in the OGC Name Type Specification for Index CRSs.[20] An Index CRS allows only integer coordinates with spacing ("resolution") of 1, hence resembling Cartesian coordinates; therefore, there is no resolution value stored.

Table 12 — IndexAxis structure

Name	Definition	Data type	Multiplicity
lowerBound	Lowest array coordinate along this axis	integer	One (mandatory)

upperBound	Highest array coordinate along this axis	integer	One (mandatory)

7.3.3 Regular Axis

Axis type RegularAxis has no restriction on the CRS used; as it is regularly spaced it contains the common distance, i.e.: resolution, as a part of the axis definition.

Table 13 — RegularAxis structure

Name	Definition	Data type	Multiplicity
lowerBound	Lowest coordinate along this grid axis	string	One (mandatory)
upperBound	Highest coordinate along this axis	string	One (mandatory)
resolution	resolution along this axis	string	One (mandatory)
uomLabel	unit of measure in which values along this axis are expressed	string	One (mandatory)

Requirement 33: https://standards.isotc211.org/19123/-/2/2/req/grid-regular/resolution

In a coverage using the *grid-regular* scheme, the resolution value in a RegularAxis coverage instantantiating class shall be a nonzero value expressed in the units of measure of this axis as defined in the CRS identified in the DomainSet srsName.

7.4 Irregular grid axis

7.4.1 Overview

This class "grid-irregular" adds coverages of irregular axis types to the General Grid Coverage introduced with class grid-regular. The concept builds upon axis types with individual characteristics, such as non-referenced, referenced-equidistant, referenced-nonequidistant, etc. from which CRSs and, hence, grids are assembled. All axis types can be combined freely in a grid. Figure 16 shows some common 2-D grid types tractable with class grid-irregular. Figure 17 shows sample situations for heterogeneous axis types in a grid (time axis is assumed to run vertically).

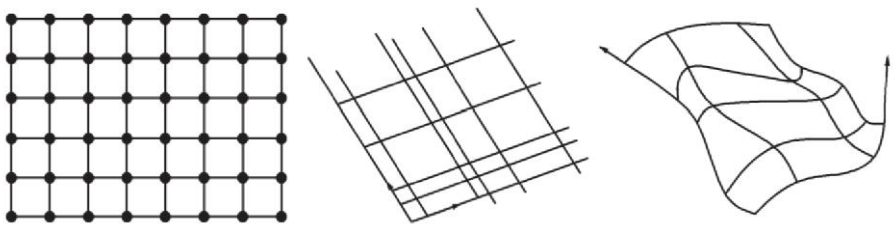

Figure 16 — Sample grid types: equidistant (left), irregular (center), displaced (right)[10]

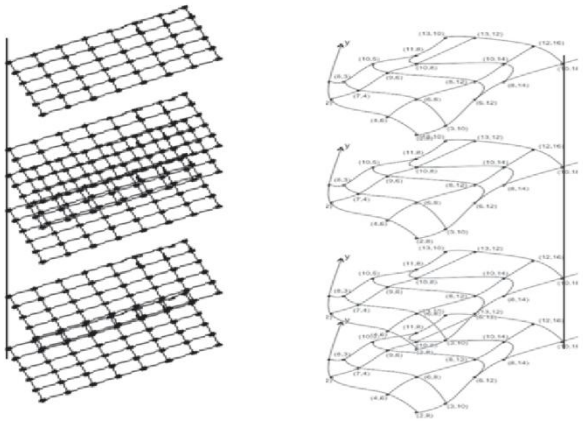

Figure 17 — Sample grid combining regular and irregular axes (left) and irregular axes and "displaced" grids (right)[10]

NOTE Skewed and rotated grids can be represented by making the grid's CRS a concatenation of any given CRS with an Engineering CRS describing, e.g., any affine transformation of the original grid.

Class grid-irregular extends class grid-regular with further axis types, hence it requires implementation of that class as a prerequisite.

Requirement 34: https://standards.isotc211.org/19123/-/2/2/req/grid-irregular/dependency

A coverage instantantiating class grid-irregular shall implement class grid-regular.

The extension consists of two cases: irregular independent grid axes (class `IrregularAxis`) and irregular correlated grid axes (class `DisplacementAxisNest`) as shown in Figure 18.

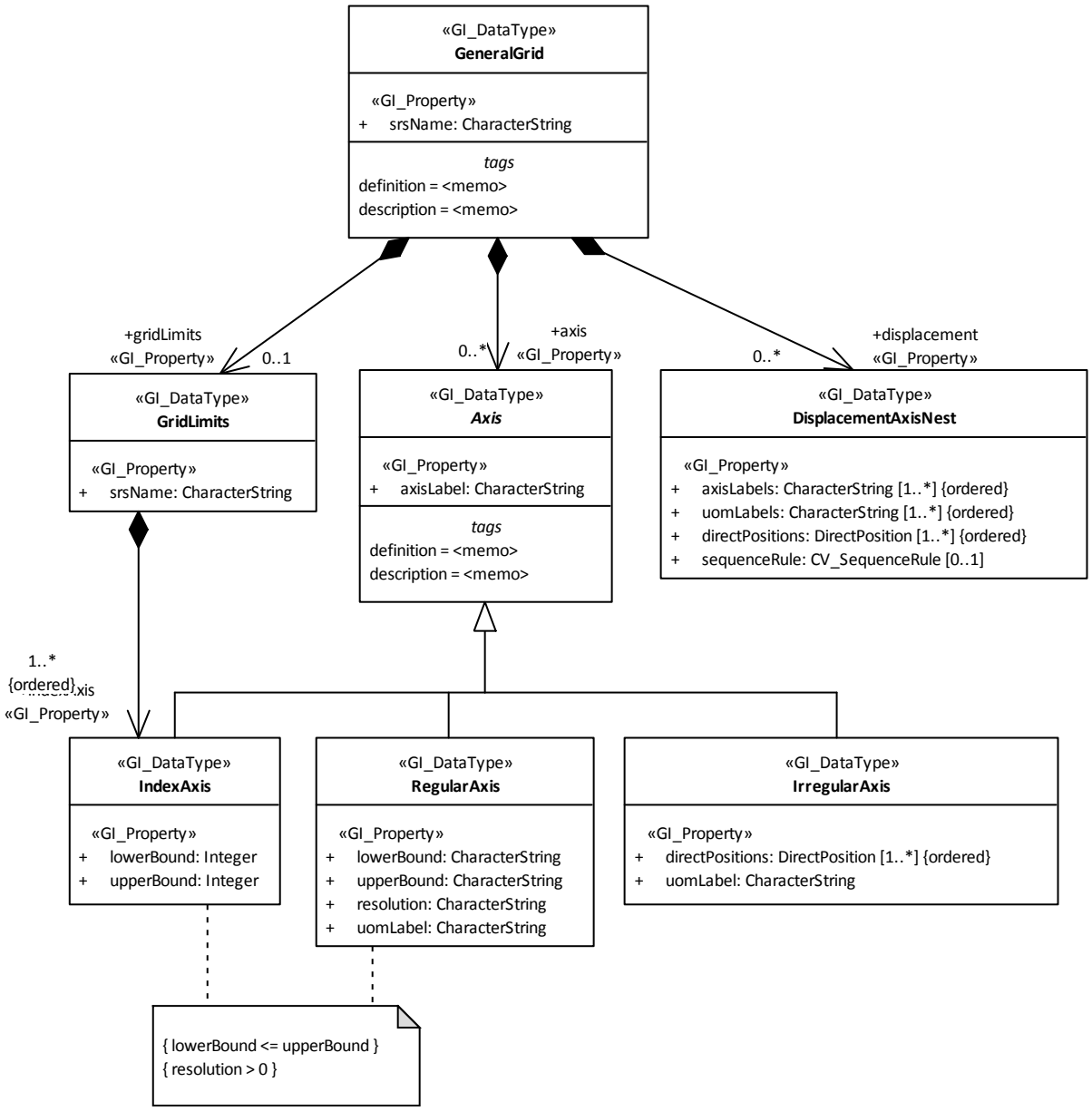

Figure 18 — `GeneralGrid` structure as per grid-irregular (overview)

7.4.2 Irregular independent grid axes

The first extension over regular axes consists of axes where spacing along an axis can have any positive increment between two neighbouring direct positions, as opposed to the constant increment of a regular axis. Therefore, all direct positions along such an axisare required to be enumerated explicitly which is achieved by replacing the lower bound / resolution / upper bound scheme by an ordered list of direct positions. Graphically, an irregular grid can be represented by straight parallel lines of varying distance for each axis (cf. Figure 16 center). Such axes are modelled by type `IrregularAxis`.

NOTE The GML 3.3 `ReferenceableGridByVector` resembles the special case that all axes are irregular, but independent. This can be modelled through a `GeneralGrid` that has only axes of type `IrregularAxis`.

Requirement 35: https://standards.isotc211.org/19123/-/2/2/req/grid-irregular/domainset

An `IrregularAxis` in the `GeneralGrid` of a coverage domain set shall conform with Figure 19 and Table 14.

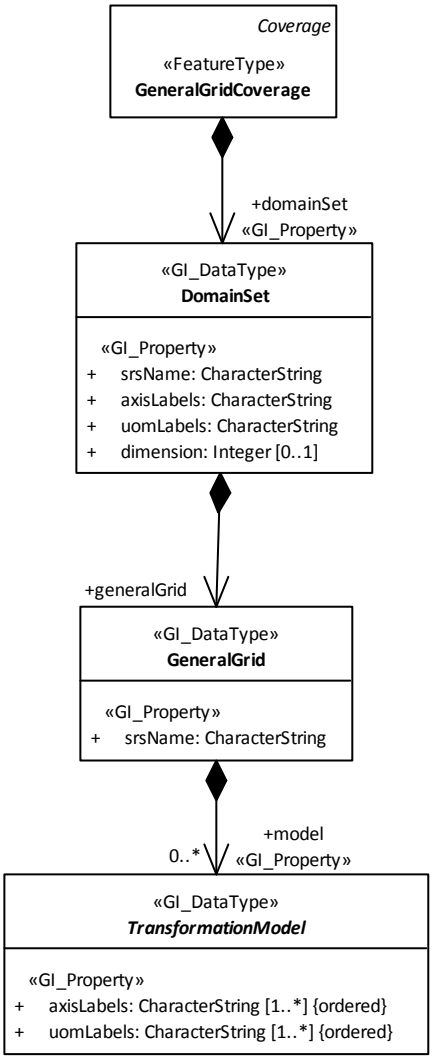

Figure 19 — `IrregularAxis` structure

Table 14 — `IrregularAxis` structure

Name	Definition	Data type	Multiplicity
`directPositions`	Ordered sequence of direct positions along this axis, without duplicates	`DirectPosition`	One or more (mandatory)
`uomLabel`	unit of measure in which values along this axis are expressed	`string`	One (mandatory)

Even in irregular axes the “grid property” shall be fulfilled: coordinates in the `directPositions` list shall be ordered, in other words: grid lines may not “cross”.

Requirement 36: https://standards.isotc211.org/19123/-/2/2/req/grid-irregular/ordered-axes

In any `IrregularAxis` in a coverage the `directPosition` values shall be listed in strictly monotonic order as defined in the CRS indicated in the `srsName`.

NOTE "Strictly monotonic" means that the sequence of position values is either completely in increasing order, or decreasing. Neither are changes in direction is allowed, nor equality of any two positions. This is to ensure that applications will not run into singularities causing, e.g., a division by zero.

7.4.3 Irregular correlated grid axes

An axis being part of a displacement grouping generalizes irregular axes further. Several axes together represent a grid where the individual direct positions of range values are situated arbitrarily in space/time (as long as the grid property is fulfilled). In such axis groups, informally called "nests", the coordinates of direct positions are not tied to the crossing points of "straight" grid lines but "displaced". Direct position coordinates, consequently, can vary freely as long as the topological neighbourhood relationship is retained. This leads to "displaced grids" as shown in Figure 16 right, keeping in mind that the auxiliary curves serve only for human understanding and do not have a technical meaning.

Not all axes in a grid need to participate in a nest, and a grid may contain several disjoint nests (although this case is unlikely). An example of a combination of a displaced nest in Lat/Long with an irregular time axis is shown in Figure 17 right.

The corresponding structure, `DisplacementAxisNest`, combines several axes to a single nest where the coordinates are enumerated individually for each direct position. Storagewise, the direct positions are no longer associated with individual axes, but collectively form an array (tensor) which is stored in the `DisplacementAxisNest` structure. The linearization scheme of this array is stated in the `sequenceRule` the same way as the linearization is described for the range set array.

NOTE In terms of storage needs, axis nests are most demanding for describing direct position coordinates: For an index axis, the constant increment of 1 does not need to be stored at all; for a regular axis a single resolution value is sufficient per axis; for an `IrregularAxis` a sequence of direct positions along each such axis is required; nests, finally, require an n-D tensor, i.e., an array which stores the coordinates of each direct position for the axes participating in the nest (cf. `DisplacementAxisNest`). With such grids, the domain set volume to be stored in a coverage can reach and even exceed the range set volume.

Table 15 — `DisplacementAxisNest` structure

Name	Definition	Data type	Multiplicity
`axisLabels`	Axes involved in the "nest" of displaced direct positions; these axes shall form a subset of the `GeneralGrid axisLabels`. List items are whitespace-separated, uom items do not contain whitespaces.	`string`	One or more (mandatory)
`uomLabels`	List of units of measure in which values along the axes are expressed, in the same order as the corresponding axes in `axisLabels`. List items are whitespace-separated, uom items do not contain whitespaces.	`string`	One or more (mandatory)
`directPositions`	Array of direct positions along this axis, linearized according to the sequence rule (see next)	`DirectPosition`	One or more (mandatory)
`coverageFunction`	Function describing the mapping from the domain to the range of the coverage	`CoverageFunction`	Zero or one (optional)

NOTE 1 Not all axes of a coverage need to participate in such a displacement nest. For example, Lat and Long can form a surface in 3-D space whereas time axis is irregular.

NOTE 2 The Annex B type `ReferenceableGridByArray` resembles the special case that all axes form one nest – in other words, for each range value its direct position is explicitly listed in the domain set. This case is reflected in CIS through a `GeneralGrid` which has only displacement axes with one array (holding the direct position coordinates) associated with all these axes.

NOTE 3 There is no monotonicity requirement on displaced axes in the way it is for irregular axes. In practice, however, it is recommended that coverage generators avoid grids that can potentially lead to issues for coverage consumers - for example, singularities like neighbouring points sharing the same (or a very close) coordinate could lead to a division by zero. Conversely, it is recommended that applications reading coverages be ruggedized to cope with borderline cases in an appropriate way.

7.5 Transformation grid

7.5.1 Transformation

Class “grid-transformation” establishes coverages with algorithmically defined grids. In class grid-regular, grids are defined through various well-known principle and (comparatively simple) computation methods. In the most general case, however, a purpose-built code – referred to in this document as a “transformation” – with a particular variable instantiation determines the grid and its direct positions.

Requirement 37: https://standards.isotc211.org/19123/-/2/2/req/grid-transformation/dependency

A coverage using the grid-transformation scheme shall implement class grid-regular.

Coverages in this class grid-transformation rely on an abstract component `TransformationModel` which is not instantiatable itself. The implementable case currently supported by this document – SensorML coverages – is defined in the next Subclause 7.5.2.

Requirement 38: https://standards.isotc211.org/19123/-/2/2/req/grid-transformation/structure

A coverage using the grid-transformation scheme shall conform with Figure 20 and Table 16.

Requirement 39: https://standards.isotc211.org/19123/-/2/2/req/grid-transformation/subclass

A coverage using the grid-transformation scheme shall implement an instantiatable subclass of (abstract class) `TransformationModel`.

NOTE It is recommended to ensure that transformations are invertible (i.e. an inverse transformation exists) in order to support the determination of the associated grid location of a given direct position.

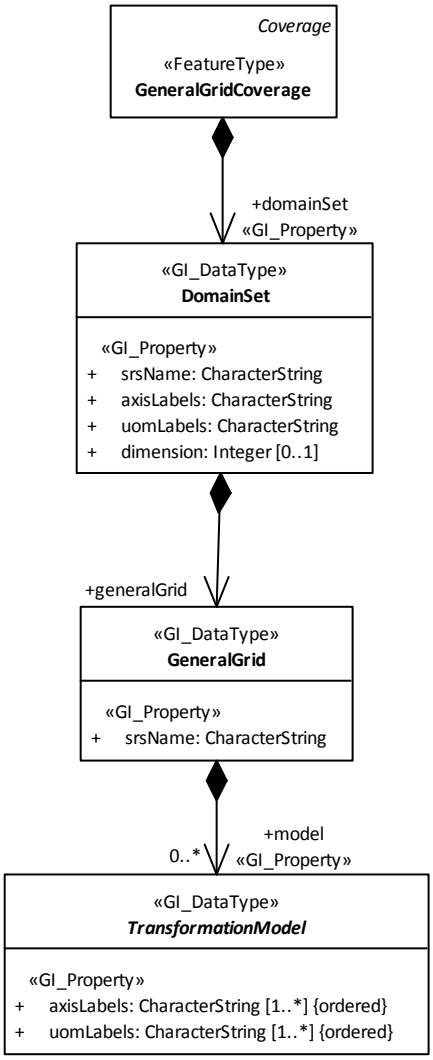

Figure 20 — `GeneralGrid` structure as per grid-transformation

Table 16 — `TransformationModel` structure

Name	Definition	Data type	Multiplicity
`axisLabels`	Ordered, whitespace-separated list of axes involved in the transformation model. Axis names do not contain whitespace.	`string`	One or more (mandatory)
`uomLabels`	Ordered, whitespace-separated list of units of measure in which values along each axis are expressed (in same sequence as in `axisLabels`). Uom items do not contain whitespaces.	`string`	One or more (mandatory)

7.5.2 SensorML

A special case of a transformation is provided by SensorML 2.0,[19] in CIS modelled through coverage type `SensorModelCoverage`. This class supports one special case of such a transformation as defined by SensorML 2.0. Such a sensor model involves two inputs: a sensor model description containing free variables plus a separate set of variable instantiations (see Table 17). As the sensor model defines the grid and its direct positions, this transformation effectively represents the coverage domain set.

Requirement 40: https://standards.isotc211.org/19123/-/2/2/req/grid-transformation/sensormodel-structure

A SensorModelCoverage shall conform with Figure 21 and Table 17.

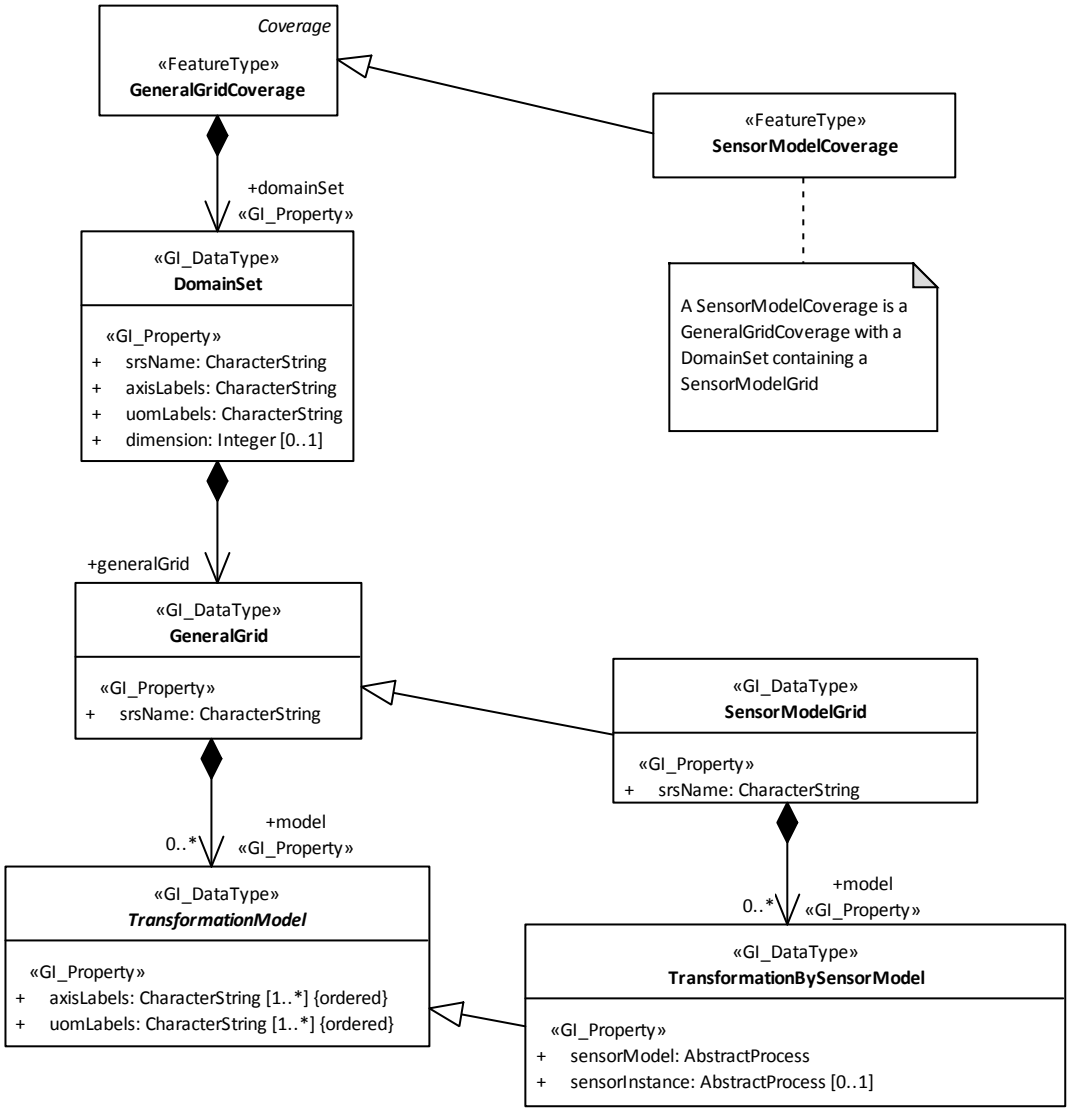

Figure 21 — `TransformationBySensorModel` structure

Table 17 — `TransformationBySensorModel` structure

Name	Definition	Data type	Multiplicity
`sensorModel`	SensorML model yielding the direct positions of the grid	`SML::AbstractProcessPropertyType`	One (mandatory)
`sensorInstance`	Parameter values for the sensor model	`SML::AbstractProcessPropertyType`	Zero or one (optional)

The `TransformationBySensorModel` of the SensorML grid inherits attributes `uomLabels` and `axisLabels` that will be a directive to the sensor model software for the computed output geo locations. In general, these attributes will have no effect whatsoever on sensor model calculations except for the last stage when the output geo locations will be transformed from the native units and CRS of the software to the specified units and CRS of the `TransformationBySensorModel`.

7.6 Number of direct positions in grid

The set of direct positions in a grid is given by the number of grid points. As every direct position holds one (atomic or composite) value, this number is equal to the number of values in the coverage's range set.

The number of elements in the range set is determined as follows. For some `GeneralGrid` g, let n_x be the number of `IndexAxis` elements, n_r the number of `RegularAxis` elements, n_i the number of `Irregular` axis elements, n_d the number of `DisplacementAxisNest` elements associated with any of the `DisplacementAxis` items, and n_t be the number of `TransformationModel` elements associated with any of the `TransformationAxis` items.

Let the following positive integer numbers be defined for the number of direct position coordinates along axes or axis combinations:

For `IndexAxis`: px_a := $g.a$.upperBound − $g.a$.lowerBound + 1 for $a \in g$.IndexAxis;

For `RegularAxis`: pr_a := trunk(($g.a$.upperBound−$g.a$.lowerBound+1)/resolution) (i.e., rounded down) for $a \in g$.RegularAxis;

For `IrregularAxis`: pi_a := card($g.a$.directPositions) for $a \in g$.IrregularAxis;

For `DisplacementAxis`: pd_d := card($g.d$.directPositions) for $d \in g$.displacement;

For `TransformationAxis`: pt_m := card($f(g)$) for $m \in g$.model where f is a function on g delivering all direct positions (such as a sensor model);

Then, the number n_p of direct positions in g is given by the product of all the above items:

$$n_p := \prod_a px_a * \prod_a pr_a * \prod_a pi_a * \prod_d pd_d * \prod_m pt_m$$

where a partial product is 1 if no axis of the corresponding type exists in the domain set.

Requirement 41: https://standards.isotc211.org/19123/-/2/2/req/grid-regular/number-of-values

The `RangeSet` of a coverage containing the above `GeneralGrid` shall contain exactly n_p value items.

NOTE This requirement applies to grid coverages in general, therefore is attached to the base class, grid-regular.

8 Multi-Curve Coverage

Multi-Curve Coverages are part of conformance class “mesh”, together with Multi-Surface Coverages and Multi-Solid Coverages.

Requirement 42: https://standards.isotc211.org/19123/-/2/2/req/mesh/dependency

A coverage instantiating class mesh shall conform with class *coverage*.

A `MultiCurveCoverage` describes a field through a set of curves, each of which carries a range value. Technically, the domain is given by a set of curves collected in a `GML::MultiCurve`.

Requirement 43: https://standards.isotc211.org/19123/-/2/2/req/mesh/multicurve-domainset

In a `MultiCurveCoverage`, the `domainSet` shall consist of a `GML::MultiCurve`.

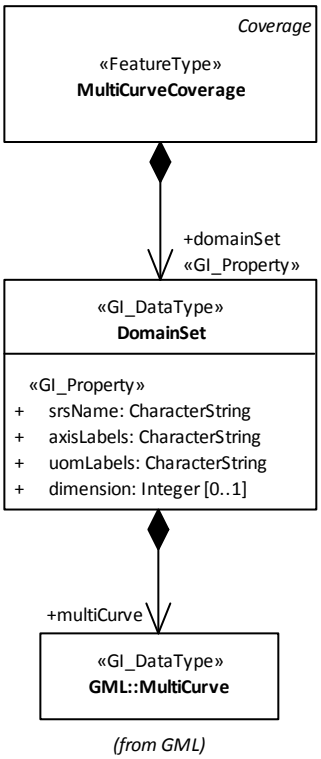

Figure 22 — UML diagram of `MultiCurveCoverage` structure

NOTE As GML `MultiCurve` elements do not carry along CRS, axis labels and uom labels as is the case with the domain set in `MultiPoint` and `GeneralGrid`, in CIS this information is currently missing until GML gets updated. It is recommended, therefore, to provide an `Envelope` and assume its CRS, axes and uom settings also for the `domainSet`.

9 Multi-Surface Coverage

Multi-Surface Coverages are part of class mesh, together with Multi-Curve Coverages and Multi-Solid Coverages.

A `MultiSurfaceCoverage` describes a field through a set of surfaces, each of which carries a range value. Technically, the domain is given by a set of surfaces collected a `GML::MultiSurface`.

Requirement 44: https://standards.isotc211.org/19123/-/2/2/req/mesh/multisurface-domainset

In a `MultiSurfaceCoverage`, the `DomainSet` shall consist of a `GML::MultiSurface`.

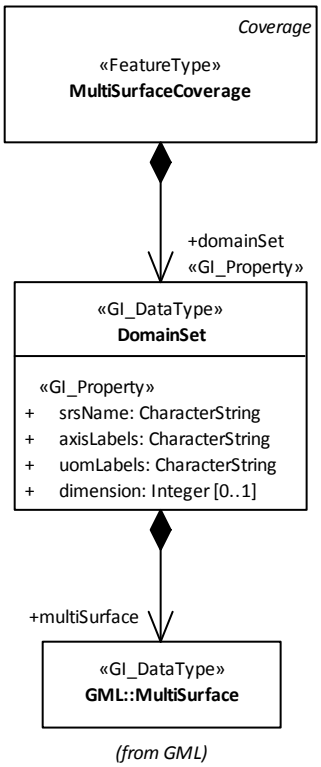

Figure 23 — UML diagram of `MultiSurfaceCoverage` structure

NOTE As GML `MultiSurface` elements do not carry along CRS, axis labels and uom labels as is the case with the domain set in `MultiPoint` and `GeneralGrid`, in CIS this information is currently missing until GML gets updated. It is recommended, therefore, to provide an `Envelope` and assume its CRS, axes and uom settings also for the `domainSet`.

10 Multi-Solid Coverage

Multi-Solid Coverages are part of class mesh, together with Multi-Curve Coverages and Multi- Surface Coverages.

A `MultiSolidCoverage` describes a field through a set of solids, each of which carries a range value. Technically, the domain is given by a set of solids collected in a `GML::MultiSolid`.

Requirement 45: https://standards.isotc211.org/19123/-/2/2/req/mesh/multisolid-domainset

In a `MultiSolidCoverage`, the `DomainSet` shall consist of a `GML::MultiSolid`.

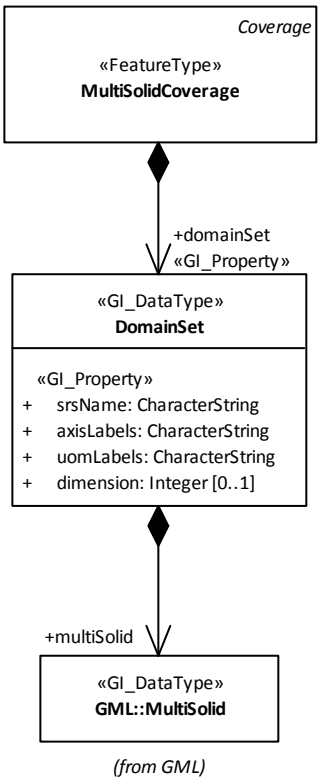

Figure 24 — UML diagram of `MultiSolidCoverage` structure

NOTE As GML `MultiSolid` elements do not carry along CRS, axis labels and uom labels as is the case with the domain set in `MultiPoint` and `GeneralGrid` in CIS this information is currently missing until GML gets updated. It is recommended, therefore, to provide an `Envelope` and assume its CRS, axes, and uom settings also for the `domainSet`.

11 Coverage partitioning

11.1 Overview

This class *partitioning* establishes an alternative representation for coverages through partitioning into sub-coverages or direct enumeration of position/value pairs.

11.2 Partitioning

With the coverage extensions provided by this class coverages can be composed from other coverages which are either copied in directly ("domain-and-range" variant), or referenced by coverage id ("partitioning" variant), or can contain single values per direct position ("position/value pair" variant, sometimes also called "geometry/value pair" or "interleaved").

Coverages embedded ("sub-coverages") can be of the same or lower dimension than the coverage embedding them ("super-coverage"). The `partition` element in the super-coverage, acting as a connection between sub- and super-coverage, contains an `envelope` element determining the sub-coverage's position relative to the super-coverage. A coverage can be part of several partitioned coverages simultaneously, thereby allowing shared regions. A partitioned coverage can itself be part of another partitioned coverage, thereby allowing trees of coverages to be built recursively.

In the position/value pair approach, single range values (which can be composite, such as RGB pixel values) are listed together with their direct position.

All of the above variants can be combined freely within a single coverage as per this standard. However, an implementation may constrain the partitioning choices available, such as to "partitioning only along time axis" or "only equi-sized sub-coverages". Further, it may support selection of partitioned and "geometry/value pair" representation.

Requirement 46: https://standards.isotc211.org/19123/-/2/2/req/partitioning/dependency

A coverage using the *coverage-partitioning* scheme shall conform to class *coverage*.

Requirement 47: https://standards.isotc211.org/19123/-/2/2/req/partitioning/structure

A coverage using the *coverage-partitioning* scheme shall conform to Figure 25, Table 18, Table 19, 0, Table 21, and 0, with `srsName`, `axisLabels`, and `uomLabels` only required in case a `PositionValuePair` component is present.

The partitioning mechanism effectively establishes a nesting of coverages. This nesting is required to be acyclic, i.e. a coverage cannot contain itself, neither directly nor indirectly.

Requirement 48: https://standards.isotc211.org/19123/-/2/2/req/partitioning/no-circular-reference

A coverage shall not reference itself through a `partition` element, neither directly nor indirectly.

The domain set of all sub-coverages participating in a partitioned coverage must lie inside the extent of the super-coverage domain set and additionally is required to fulfil homogeneity criteria to ensure that the resulting structure adheres to the definition of a coverage.

A coverage can act as sub-coverage in more than one coverages.

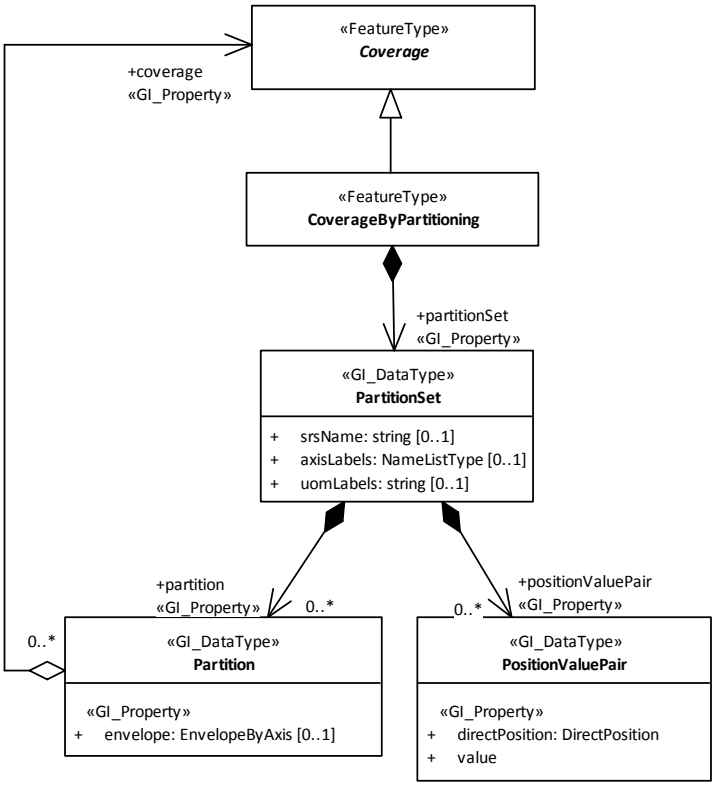

Figure 25 — UML diagram of `CoverageByPartitioning` structure as per *coverage-partitioning* (class-specific part only, see Figure 3 for the complete structure)

Table 18 — `CoverageByPartitioning` structure

Name	Definition	Data type	Multiplicity
`partitionSet`	Set of coverages or single positioned values which together make up the coverage on hand, containing at least one `partition` or at least one `value`.	`PartitionSet`	one (mandatory)

Table 19 — `PartitionSet` structure

Name	Definition	Data type	Multiplicity
`partition`	Sub-coverage being part of the coverage on hand, together with positioning information	`Partition`	Zero or more (optional)
`value`	Range value being part of the coverage on hand, together with positioning information	`PositionValuePair`	Zero or more (optional)
`srsName`	Identifier of the CRS in which the coverage domain set coordinates are expressed	`string`	Zero or more (optional)
`axisLabels`	List of whitespace-separated pairwise distinct axis names, each one corresponding to exactly one axis in the CRS, matching the position in the axis name list and the axis position in the CRS. Axis names do not contain whitespace. They can be identical to the respective CRS axis abbreviation, but do not have to.	`string`	Zero or more (optional)
`uomLabels`	List of whitespace-separated units of measure (uom), where each uom belongs to the axis matched by the position in `axisLabels`. Uom items do not contain whitespaces.	`string`	Zero or more (optional)

NOTE A partitionSet, as indicated, is required to contain at least one element. A mix of larger partitions and single values is possible.

Table 20 — `Partition` structure

Name	Definition	Data type	Multiplicity
`envelope`	Envelope of sub-coverage making up this partition; default: envelope of the coverage referenced	`EnvelopeByAxis`	Zero or one (optional)
`coverage`	Coverage acting as partition , directly stored here or through some resolvable reference, such as coverage id or a URL	`Coverage`	One (mandatory)

Table 21 — `PositionValuePair` structure

Name	Definition	Data type	Multiplicity

directPosition	Direct position of the coverage to which value is assigned	DirectPosition	One (mandatory)
value	Coverage value to be associated with directPosition	any	One (mandatory)

Sub-coverages can be stored directly as the value of coverage, or they can be given as a reference, such as coverage id or a URL.

NOTE Support for these alternatives can vary across data format encodings. Further, as this is a requirement which a server is required to fulfil, an implementation can potentially restrict the options for referencing coverages to those ones where it can control this acyclicity requirement.

11.3 CRS and partition envelope constraints

The sub-coverage CRS is required to allow the coverage data to be embedded in the super-coverage referencing it.

Requirement 49: https://standards.isotc211.org/19123/-/2/2/req/partitioning/axis-subset

In a coverage *s* with CRS *cs* contained as partition in a coverage *c* with CRS *cc* the following shall hold: *cs* is obtained from *cc* by deleting zero or more (but not all) axes from *cc*.

NOTE 1 This definition, among others, enforces an identical axis order among those axes present in both the sub- and super-coverage CRSs.

EXAMPLE A timeseries datacube with CRS axes *Lat/Long/t* can contain sub-coverages whose CRS axes are given by *Lat/Long*, but not by *Long/Lat*. A datacube with axis order *t/Lat/Long* likewise can contain sub-coverages with a *Lat/Long* CRS.

Lower-dimensional sub-coverages are embedded as slices of thickness one into the super-coverage.

Requirement 50: https://standards.isotc211.org/19123/-/2/2/req/partitioning/slice

In a coverage *s* with CRS *cs* contained as partition in a coverage *c* with CRS *cc* the following shall hold: lowerBound = upperBound in the envelope of *c* for all axes not occurring in *cc* but occurring in *cs*.

NOTE 2 This allows to "lift" coverage parts into higher-dimensional spaces in the super-coverage, such as embedding a 2-D Lat/Long timeslice into a 3-D Lat/Long/time datacube.

The partitionEnvelope element does not need to repeat coordinate axis values of the sub-coverage if they are identical in the context of the super-coverage.

Requirement 51: https://standards.isotc211.org/19123/-/2/2/req/partitioning/bounds

For any axis of the domain set CRS *cc* of some coverage *c* containing some coverage *p* as a partition, any axis not listed in *c*'s partitionEnvelope within *p* the default lowerBound and upperBound of this axis in the partitionEnvelope shall be given by the corresponding values in the DomainSet of *p*.

NOTE 3 Axis identification and sequence is unambiguous even when axes are left out because partitionEnvelope coordinates are expressed in terms of the super-coverages CRS which defines all axes and their sequence.

11.4 Domain set constraints

The sub-coverage domain sets, as well as single direct positions, is required to be non-overlapping (considering all axes plus the range components) and properly contained in the super-coverage; missing boundary values are represented as a null value.

NOTE 1 Such null values can be used whenever the actual extent of the super-coverage is not known in the super-coverage itself, such as in timeseries where further timeslices can be appended at any time. The representation of such a null value is defined in the concrete encodings.

Requirement 52: https://standards.isotc211.org/19123/-/2/2/req/partitioning/ignore-nulls

For any coverage *p* referenced as `partition` in a coverage *c*, the `envelope` of *p* shall be a subset of the domain set of *c*, obtained by ignoring all values of `lowerBound` and `UpperBound` in the `envelope` of *c* which have a null value.

Requirement 53: https://standards.isotc211.org/19123/-/2/2/req/partitioning/disjoint-extents

For any coverage *c* of type `CIS:CoverageByPartitioning`, all `partition` and `value` components shall have pairwise disjoint extents across any of its range components.

EXAMPLE Band-interleaved (BIL) representation can be achieved through multiple sub-coverages all registered to the same extent, but each one adding an individual band.

Requirement 54: https://standards.isotc211.org/19123/-/2/2/req/partitioning/have-null-for-regions

In a coverage containing at least one direct position for which no value is stored there shall be at least one null (i.e., nil) value defined in its range type, for use at such direct positions.

NOTE 2 Such "undefined areas" can only occur with coverages containing partitions (in a domain / range representation there must always exist a value for each direct position). This rule ensures that "null values" exist when needed.

NOTE 3 Such "default" null values can differ among direct positions, and an implementation is free to choose values non-deterministically. It is good practice, though, to use a single value whenever possible.

11.5 Range type constraints

Sub- and super-coverage are required to have compatible range types – either identical ones, or partitions contribute parts of the full super-coverage range component record.

Requirement 55: https://standards.isotc211.org/19123/-/2/2/req/partitioning/rangetype-subset

For any coverage *p* with range type *rp* referenced as a `partition` in a coverage *c* with range type *rc*, the following shall hold: *rp* is obtained from *rc* by deleting zero or more range components from *rc*.

NOTE Sub-coverage bands are visible in the super-coverage under the name indicated in the range type translation list, which cannot lead to name clashes in the super-coverage (i.e. range component names still have to be pairwise distinct). Further, from the super-coverage perspective, all range components "imported" are required to adhere to the same range type definition to avoid violating the basic definition of range type coherence in a coverage.

EXAMPLE Band-interleaved storage of satellite imagery, as well as variables in climate model output can be accomplished this way: single bands, or combinations of bands, can go into separate sub-coverages which are linked together through a super-coverage.

If the partitions altogether are not commensurate to the complete range type structure then the range components not covered are equivalent to some null value (which needs to be defined in this case).

Requirement 56: https://standards.isotc211.org/19123/-/2/2/req/partitioning/have-null-for-range-components

In any coverage containing at least one range component for which no value is stored there shall be at least one null (i.e., nil) value defined in the corresponding range type component, for use at such range values.

EXAMPLE 1 Consider an RGB coverage where the colour bands are factored out into partitions. Assume that there are only partitions for the red and green, but not for the blue band. In this case, the range type definition of the RGB coverage is required to provide a null value for the blue band so that an equivalent "flat" coverage can be constructed which contains null values in all direct positions for the missing blue band.

EXAMPLE 2 Band interleaving combined with spatial partitioning (such as in mosaics) can lead to small islands of null values. For each of them, a proper null value definition is required to exist, allowing an implementation to interpret the missing value as one of these null values.

12 Coverage encodings

12.1 Overview

Coverages can be encoded in many suitable formats, some of which are addressed in this document.

Not all encodings are able to represent the full information making up a coverage, i.e.: not all encodings are informationally complete. Common restrictions include range type details which, for example in GeoTIFF, cannot be transported properly. This can affect choice of the encoding used.

12.2 XML

12.2.1 General

Requirements class *xml-coverage* establishes how coverages are represented in the XML encoding format. This is done through the XML schema included in this document.

Requirement 57: https://standards.isotc211.org/19123/-/2/2/req/xml-coverage/dependency

A coverage using the *xml-coverage* scheme shall implement class *coverage*.

Requirement 58: https://standards.isotc211.org/19123/-/2/2/req/xml-coverage/validate

Any XML document instantiating one of `MultiPointCoverage`, `GeneralGridCoverage`, `MultiCurveCoverage`, `MultiSurfaceCoverage`, `MultiSolidCoverage` or a subtype thereof shall validate against the XML schema and Schematron rules provided with this standard.

NOTE 1 The XML Schema included in this document contains several examples for different coverage instances encoded in XML.

NOTE 2 While the CIS XML Schema uniformly uses CamelCase element names with the first character capitalized, GML uses CamelCase with the first character of the element name sometimes upper and sometimes lower case.

NOTE 3 Normally, tag names are self-explanatory. In some cases, however, it is important to keep them short due to the high number of occurrences. This is in particular the case with enumerations of direct positions and position/value pairs. Hence, the following abbreviations have been chosen: C for coordinate, V for value, CV for composite range value, PV for position/value.

NOTE 4 The SWE Common XML schema for DataRecord, used by the coverage range type, has been copied verbatim from OGC 08 094r1 into the XML schema accompanying this document, to make the coverage definition more self-contained and convenient for the reader.

12.2.2 Relation with GML

While historically based on ISO 19136-2:2020[11][12] aka OGC GML, in this document, the XML encoding of coverages has undergone careful corrections in the structuring, and additional features and coverage types have been added, most notably the General Grid Coverage. Additionally, the necessary alignment with ISO 19123-1:2023 has been established.

To make the XML schema more lightweight and self-contained, several GML definitions have been migrated into the coverage schema, at the same time simplifying these very general definitions for the

particular use with coverages. Further, highly repetitive elements have been given particularly short names to keep file size low. Therefore, this XML conformance class is not a GML Application Profile in the sense of the GML standard, ISO 19136-2:2020.

In parallel to this, the following naming convention has been adopted for *xml-coverage*: Element and type names are in camel case with first letter capitalized; attribute names are in camel case with first letter lowercase. This is a change compared to ISO 19136-1:2020 where both lower and upper case can appear in element names, depending on their role in the schema. The reason for this change is to achieve coherent upper/lower case conventions across the XML and JSON encoding of CIS as well as to simplify XML handling towards common XML Schema practices.

In GML,[11] all coverage types are derived from the abstract `GML::Coverage` data type containing a `DomainSet` and a `RangeSet` component. The Coverage Implementation Schema extends this with two additional components, a mandatory `RangeType` and optional `metadata`, an extensible slot for individual, application-specific metadata structures.

In summary, the following CIS changes apply in relation to GML:

— The object/property model of GML is abandoned, leading to a substantial reduction of complexity.

— There are several extra concepts not present in GML, ranging from model (grid definition by axis rather than by grid type, SensorML domains, etc.) over representation (partitioning and geometry / value pairs) to encoding (addition of JSON).

— Coordinates are not required to be numeric only, but can also contain strings such as ISO 8601 series date / timestamps or categorical values. This is instrumental for general multi-dimensional coverages.

— A point cloud coverage type, `MultiPointCoverage`, is provided which semantically is equivalent to GML and ISO 19123-2:2018, but allows string coordinates as described above. The same step has not been done for `MultiCurve/Surface/SolidCoverage` in this edition of this document due to the complexity involved.

Further changes which SensorML introduces in relation to GML are detailed in Reference [22].

Concerning ISO 19136-2:2015[12] aka OGC GML 3.3[38]: as GML 3.3 and ISO 19123-2:2018 / OGC CIS 1.0 are both “leaf packages” of GML 3.2.1, such that there is no direct dependency of one on the other, use of the GML 3.3 referenceable grid elements in the `GML::DomainSet` of a `ReferenceableGrid-Coverage` violates OGC Modular Specification[23] Requirement 24 in that GML 3.3 is not a conformant extension to CIS 1.0.

12.3 JSON Coverage

Class *json-coverage* establishes how coverages are represented in the JSON encoding format.

Requirement 59: https://standards.isotc211.org/19123/-/2/2/req/json-coverage/dependency

A coverage using the *json-coverage* scheme shall implement class *coverage*.

Requirement 60: https://standards.isotc211.org/19123/-/2/2/req/json-coverage/rfc7159

A coverage encoded in JSON shall conform to IETF RFC7159.

Requirement 61: https://standards.isotc211.org/19123/-/2/2/req/json-coverage/validate

Any JSON document instantiating a concrete subtype of Coverage shall conform to the JSON Schema definitions being part of this standard.

NOTE 1 The JSON Schema directory included in this document contains several examples for different coverage instances encoded in JSON.

NOTE 2 The SWE Common JSON schema for DataRecord, used for the coverage range type, has been copied verbatim from OGC 24-014 into the XML schema accompanying this document, to make the coverage definition more self-contained and convenient for the reader.

12.4 Multipart encoding

12.4.1 Overview

Class “multipart” establishes how coverages can be packaged into multiple files, meaning that the coverage document (henceforth referred to as the “first part”) has one or more components shifted out into separate documents (henceforth called “further parts”). To maintain connection between the parts, the first part references all other parts through URLs (which may be local). Packaging can be done through any appropriate container format. Additionally, parts can be stored outside the package, referenced by URLs.

NOTE 1 Among the suitable container formats are multipart MIME,[2] GMLJP2, zip, and tar. Out of those, MIME is normatively defined here.

Such a splitting is particularly useful for the range set so as to allow a different, possibly more efficient encoding of this (typically) bulk of information. However, with the same argument other parts of the coverage (such as a large domain set with displaced axes) can be shifted into further parts as well.

To achieve a complete representation of the coverage, the encoding used in the first part needs to be “informationally complete”, i.e. able to hold the complete coverage information. Further, it needs to allow expressing references (which replace the substructure – such as the range set – to be shifted into a separate part). Notably, the format used in the further parts does not need to be informationally complete with respect to coverage metadata; however, it needs to be able to represent the values factored out of the first-part document.

NOTE 2 Among the list of suitable formats for the first part are GML and JSON. Image/data formats like GeoTIFF and netCDF are suitable formats for the further parts.

Requirement 62: https://standards.isotc211.org/19123/-/2/2/req/multipart/dependency

A coverage using the multipart scheme shall implement class *coverage*.

Requirement 63: https://standards.isotc211.org/19123/-/2/2/req/multipart/coverage

A coverage encoded as a multipart MIME message shall adhere to IETF RFC 2387[2] in that it consists of a multipart MIME document with a `Content-Type` parameter of value “`Multipart/Related`” and a `Type` parameter containing a MIME type identifier matching the encoding of the first (“root”) part; references to further parts located in the same container as the first-part coverage shall use a local "cid" (Content-ID) URL as specified by IETF RFC 2392[5].

EXAMPLE The MIME type identifier of GML is “application/gml+xml”.

12.4.2 Root part

The *root part* of a multipart coverage consists of the top-level structure of the coverage. Each container format needs to individually determine how this root part is represented.

EXAMPLE In Multipart / MIME, this is the first item in the stream.

Requirement 64: https://standards.isotc211.org/19123/-/2/2/req/multipart/root

In a coverage encoded as per class multipart, the root part shall be a complete coverage as per this standard, but with one or more components replaced by a reference to the further parts of the

multipart message where these components replaced get manifested.

EXAMPLE In a GML encoded coverage, a reference can be expressed through a `fileReference` element.

NOTE Each part of the message can be encoded individually and independently, in different formats.

Requirement 65: https://standards.isotc211.org/19123/-/2/2/req/multipart/references

In a coverage encoded as per class multipart, references from the first message part (containing the coverage root part) to subsequent parts shall use the method foreseen by the container format to achieve an unambiguous identification of the further parts located in the same container as the first-part coverage.

NOTE 1 Generally, syntax and semantics of the reference may depend on the environments in which the coverage containing the reference, on the one hand, and the item referenced, on the other hand, reside: in a multipart MIME message, this will be cid identifiers; in a zip file, identification will be done through file names and paths relative to the zip directory root; this hierarchical scheme would allow relative references. In a GMLJP2 file, identification will be done through XML identifiers, i.e., locally unique gml:id attributes. If keeping a sandboxed environment is important (e.g. for security reasons), the W3C `app:` URI scheme[36] can be used.

NOTE 2 A reference can be temporarily or permanently unresolvable. In case of an unresolvable reference, the coverage can still be reconstructable through other means – for example, treatment of CRSs given by some well-known URI may be hardwired in an application handling coverages.

12.4.3 Further parts

The root part can, instead of containing coverage constituents verbatim, shift such constituents into subsequent parts of the multipart document and reference them.

Requirement 66: https://standards.isotc211.org/19123/-/2/2/req/multipart/complete

In a coverage encoded as per class multipart, any part referenced from the root part shall contain the complete information required to substitute the reference and altogether obtain a complete coverage as per the coverage type to be represented.

NOTE In earlier editions of this document, only one extra part was foreseen exclusively for the range set. Starting with this present edition, more than one coverage component can be extracted into a separate part. Besides the (often large) range set, another candidate for a separate part is the domain set in a coverage with displaced axes, as such a domain set can become just as large as the range set. In a Multi-Point Coverage, for example, the domain set often is even larger than the range set.

Annex A
(normative)

Abstract test suite

A.1 Overview

This annex specifies an abstract test suite which shall be passed in completeness by any implementation claiming conformance with this coverage implementation schema.

The test approach conceptually consists of two steps:

— Transcode the coverage from its original format into one of the formats directly addressed by this standard, following the mapping rules defined for the particular original format on hand.

— Perform all conformance tests on this transcoded coverage representation. Tests fail/succeed if they fail/succeed, respectively, on this transcoded representation.

A concrete test implementation may choose a different strategy (such as for efficiency reasons) as long as the tests behave as indicated in this abstract test suite.

The root URI for all conformance tests is https://standards.isotc211.org/19123/-/2/2/conf.

A.2 Conformance Test Class: coverage

Reference: All normative statements in class *coverage*

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

A.3 Conformance Test Class: pointcloud

Reference: All normative statements in class *pointcloud*

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

A.4 Conformance Test Class: grid-regular

Reference: All normative statements in class *grid-regular*

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

A.5 Conformance Test Class: grid-irregular

Reference: All normative statements in class *grid-irregular*

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

A.6 Conformance Test Class: grid-transformation

Reference: All normative statements in class *grid-transformation*

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

A.7 Conformance Test Class: mesh

Reference: All normative statements in class mesh

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

A.8 Conformance Test Class: partitioning

Reference: All normative statements in class partitioning

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

A.9 Conformance Test Class: xml-coverage

Reference: All normative statements in class xml-coverage

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

A.10 Conformance Test Class: json-coverage

Reference: All normative statements in class json-coverage

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

A.11 Conformance Test Class: multipart

Reference: All normative statements in class multipart

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

Annex B
(normative)

Rectified and Referenceable Grid Coverages

B.1 Abstract GML-based coverage

This Annex integrates coverage types `RectifiedGridCoverage` of ISO 19123-2:2018 and `ReferenceableGridCoverage` of OGC 16-083r3 [37]. The corresponding conformance class is *rectified+referenceablegridcoverage* (identifying URI: https://standards.isotc211.org/19123/-/2/2/conf/rectified+referenceablegridcoverage). with two separate conformance classes *rectifiedgridcoverage* (identifying URI: https://standards.isotc211.org/19123/-/2/2/conf/rectifiedgridcoverage) and *referenceablegridcoverage* (identifying URI: https://standards.isotc211.org/19123/-/2/2/conf/referenceablegridcoverage) for each of the both coverage types.

These definitions remain active and available, although it is recommended to use `GeneralGrid-Coverage` (Clause **Error! Reference source not found.**) instead which has more expressive power and modelling clarity.

Reasons why these classes have been factored out into an Annex are the following, among others:

— The general structure of the coverage class hierarchy is different, as a more natural approach has been adopted in Clause 5 through 10 (sorting by coverage dimensions) which is also consistent with the coverage fundamentals (ISO 19123-1[10]).

— Coverage types in Annex B still rely completely on GML which has some known shortcomings in comparison with the `GeneralGridCoverage` class (for example, the GML limitation to numerical coordinates which excludes date/timestamps).

— ISO 19123-2:2018 creates a tight coupling of the logical (UML) and physical (GML) level, which is overcome with this document and its clear separation of level. Adjusting the original coverage type descriptions would have required a major rewrite of the specification, without worthwhile advantages but a risk of introducing incompatibilities for existing implementations.

XML schemas together with examples are available as separate accompanying files.

B.2 Abstract GML-based coverage

Abstract class *rectified+referenceablegridcoverage* is derived from the abstract `GML::Coverage` data type in ISO 19136-1:2020 GML. This structure contains a `domainSet` describing the coverage's domain and a `rangeSet` component containing the range values ("pixels", "voxels") of the coverage. Conformance class extends `GML::Coverage` with two new components, `rangeType` and `metadata`.

— The `rangeType` element describes the coverage's range set data structure. A range value often consists of one or more fields (in remote sensing also referred to as *bands* or *channels*), however, much more general definitions are possible. Range value structure description is based on the OGC SWE Common [OGC 08-094r1] `DataRecord`.

— The abstract coverage definition is augmented with an extensible slot for metadata. The intended use is to define concrete metadata structures and their semantics in extensions or application profiles.

Figure B.1 gives an overview on coverage structure overall.

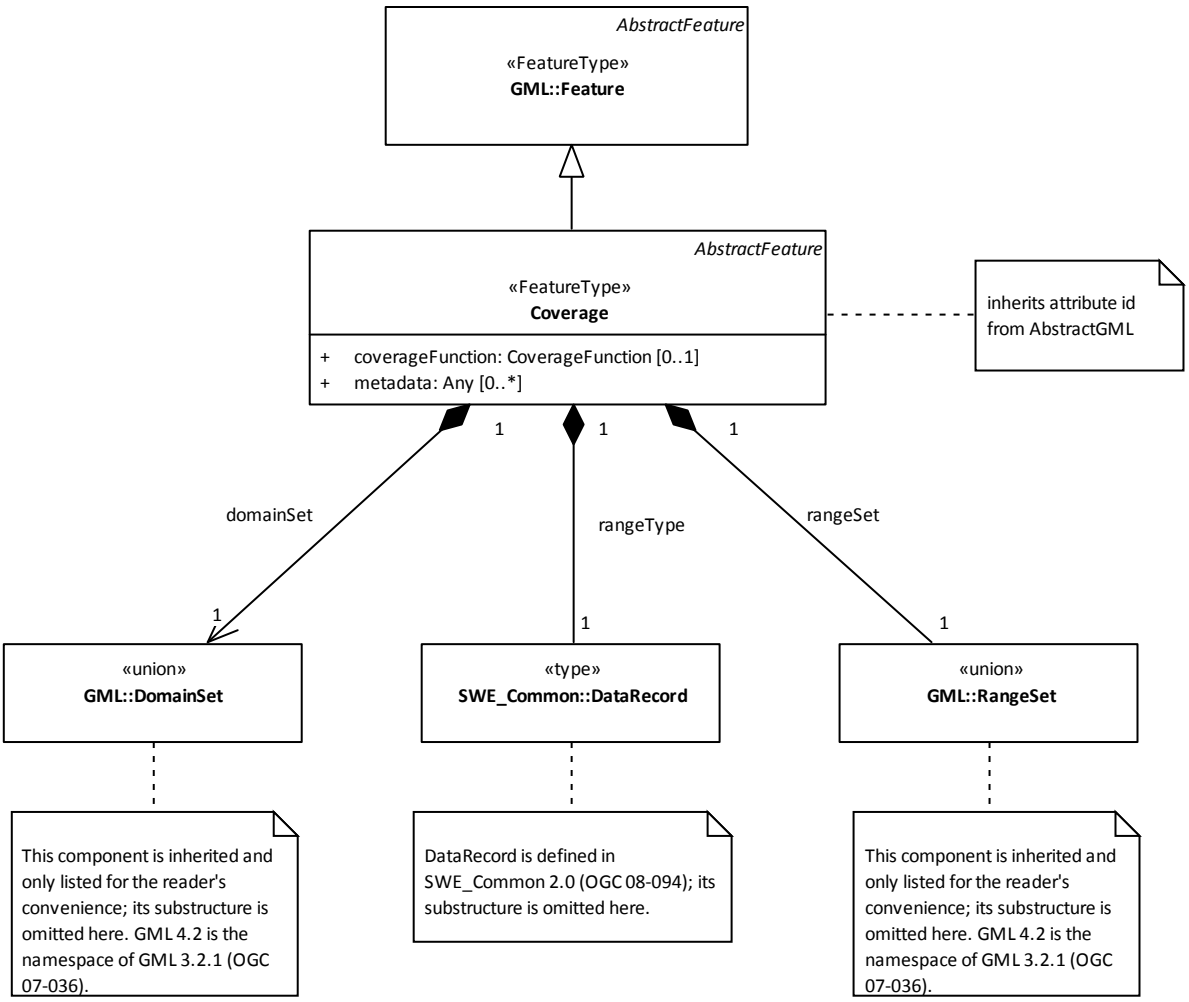

Figure B.1 — General coverage structure used in this Annex

Requirement 67: https://standards.isotc211.org/19123/-/2/2/req/rectified+referenceablegridcoverage/valid

Any instance of an instantiatable subtype of `Coverage` shall conform with the UML diagram in Figure B.1, with Table B.1, with the conformance tests defined in Annex B.5, and with the XML schema defined as part of this document.

Table B.1— `Coverage` structure

Name	Definition	Data type	Multiplicity
`coverage-Function`	GML 3.2.1 coverage function to describe how range values at coverage locations can be obtained	`GML:: Coverage-Function`	Zero or one (optional)
`metadata`	Application specific metadata	Any	Zero or more (optional)
`domainSet`	GML 3.2.1 Definition of coverage domain	`GML:: DomainSet`	One

			(mandatory)
rangeType	Structure definition of the coverage range values	SWE:: DataRecord	One (mandatory)
rangeSet	GML 3.2.1 Coverage range values	GML:: RangeSet	One (mandatory)

NOTE 1 UML data type Any is used here with the same meaning as XML's xsd:any, which does not have a direct equivalent in UML.

NOTE 2 Following the GML pattern described in Reference [14] on GML level, SWE::DataRecord is linked to rangeType via an association SWE::DataRecordPropertyType.

The coverageFunction component is identical in its syntax and meaning to the coverageFunction element defined in ISO 19136-1:2020[14], Subclause 19.3.11.

The metaData component is a carrier for any kind of application dependent metadata. Hence, no requirements are imposed here.

The rangeType component adds a structure description and technical metadata required for an appropriate (however, application independent) understanding of a coverage. For this structure description, the SWE Common DataRecord is used.

Requirement 68: https://standards.isotc211.org/19123/-/2/2/req/rectified+referenceablegridcoverage/datarecord

The range type of a coverage shall conform with the DataRecord of SWE Common as defined in OGC 08-094r1.

NOTE 3 Following GML patterns, the swe:DataRecord is linked into gmlcov:AbstractCoverage-Type via swe:DataRecordPropertyType.

Atomic data types available for range values are those given by the SWE Common data type AbstractSimpleComponent. As a range structure contains only structure definitions, but not the values themselves (these sit in the coverage range set component), the optional AbstractSimpleComponent component value is suppressed in coverages.

Requirement 69: https://standards.isotc211.org/19123/-/2/2/req/rectified+referenceablegridcoverage/no-value

For all OGC 08-094r1 SWE Common AbstractSimpleComponent subtypes in a range type structure, instance multiplicity of the value component shall be zero.

NOTE 2 Following [OGC 08-094r1], omission of the value component implies that in a DataArray there is no encoding component either.

Range values can be structured as records or arrays. Both structuring principles can be nested (and mixed) to any depth for a concrete coverage range structure definition.

Requirement 70: https://standards.isotc211.org/19123/-/2/2/req/rectified+referenceablegridcoverage/record-or-array

Wherever the SWE Common XML schema allows an AbstractDataComponent in a coverage range structure the concrete instance shall be one of the AbstractDataComponent subtypes DataRecord and DataArray.

NOTE 3 In particular, these AbstractDataComponent subtypes are *not* allowed in range structures: DataChoice (it violates coverage homogeneity), Vector, Matrix (both can be emulated by DataArray).

Within a `DataRecord` in a range structure, each of its record components is locally uniquely identified by the record component's `field` attribute.

Both `domainSet` and `rangeType` describe the coverage values given in the `rangeSet`. Hence, consistency must be enforced between them. The pertaining requirements are listed in Subclause 6.6.

Requirement 71: https://standards.isotc211.org/19123/-/2/2/req/rectified+referenceablegridcoverage/one-range-value

For each direct position in the domain set of a coverage there shall exist exactly one range value in the coverage's range set.

Both duplicates and values omitted are not allowed. For range values not known for some reason nil values can be used.

Requirement 72: https://standards.isotc211.org/19123/-/2/2/req/rectified+referenceablegridcoverage/range-validity

All range values contained in the range set of a coverage shall be consistent with the structure description provided in its range type.

In the following subclauses, `RectifiedGridCoverage` and `ReferenceableGridCoverage` are established (Figure B.2). These coverage types may be used as is, or new coverage types may be constructed by using or deriving from `Coverage` or one of its subtypes.

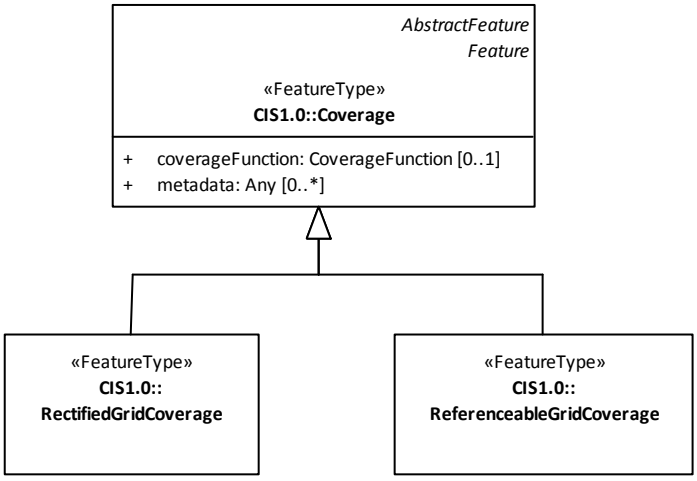

Figure B.2 — Coverage hierarchy for `RectifiedGridCoverage` and `ReferenceableGridCoverage` (overview)

B.3 Rectified grid coverages

Class *rectifiedgridcoverage* defines rectified grid coverages derived from GML.

Requirement 73: https://standards.isotc211.org/19123/-/2/2/req/rectifiedgridcoverage/dependency

A `RectifiedGridCoverage` instance shall be a subtype of `AbstractCoverage` as per class *rectified+referenceablegridcoverage*.

A `RectifiedGridCoverage` is based on a rectified grid where along each axis the direct positions have the same, constant distance (often called resolution).

Requirement 74: https://standards.isotc211.org/19123/-/2/2/req/rectifiedgridcoverage/domainset

The domain set of a `RectifiedGridCoverage` shall be a `GML::RectifiedGrid` geometry conforming with the XML schema defined as part of this document

B.4 Referenceable grid coverages

Class *referenceablegridcoverage* defines referenceable grid coverages derived from GML. `ReferenceableGridCoverage` is based on a non-rectified grid where along each axis the direct positions have an individual distance each.

NOTE 1 The working definition of a sensor model assumed is: A sensor model is a mathematical model for estimating geolocations from data recorded by a remote sensing system.

NOTE 2 This `ReferenceableGridCoverage` is a verbatim adoption of the OGC ReferenceableGrid-Coverage Extension standard,[24] an Application Profile of OGC CIS.

This class supports the following GML elements:

— `ReferenceableGridByArray`, `ReferenceableGridByVectors`, and `Referenceable-GridByTransformation` are adapted from the GML 3.3.[38] There is no dependency on GML 3.3.

— `ReferenceableGridBySensorModel` provides access to grid transformations via SensorML 2.0, where the associated SensorML 2.0 documents are recommended to be based on profiles associated with referenceable grid transformations, such as ISO 19130:2010.[15] Sensor models are used to represent referenceable grid transformations for any relevant remote sensing system.

Requirements are stated on `ReferenceableGridCoverage` instance documents that contain in their domain sets one or more referenceable grid elements, or referenceable grid elements derived thereof.

Requirement 75: https://standards.isotc211.org/19123/-/2/2/req/referenceablegridcoverage/dependency

A `ReferenceableGridCoverage` instance shall be a subtype of `Coverage` as per class *rectified+referenceablegridcoverage*.

Requirement 76: https://standards.isotc211.org/19123/-/2/2/req/referenceablegridcoverage/domainset

The domain set of a `ReferenceableGridCoverage` shall be a subtype of `AbstractReferenceableGrid` conforming with the XML schema defined as part of this document.

Below, several instantiatable subtypes of `AbstractReferenceableGrid` are introduced in turn: `ReferenceableGridByVectors`, `ReferenceableGridByArray`, and `ReferenceableGridByTransformation`, and `ReferenceableGridBySensorModel`. Their domain set is defined via a referenceable grid transformation from locations on the grid to coordinates in an external coordinate reference system.

`ReferenceableGridByVectors` defines a referenceable grid by specifying an origin and a set of offset vectors, with multiplicative coefficients that scale the offset vectors to generate a (potentially) irregularly-spaced grid.

NOTE 3 `ReferenceableGridByVectors` generalizes the mechanism used for the RectifiedGrid of GML 3.2.1, which similarly uses offset vectors but in a much more restrictive way. For a RectifiedGrid, each offset vector is always aligned with a single grid direction, while for a `ReferenceableGridByVectors` such a restriction does not hold in general.

Requirement 77: https://standards.isotc211.org/19123/-/2/2/req/referenceablegridcoverage/by-vector

A `ReferenceableGridByVectors` shall be defined by Table B.2, Table B.3, and the XML Schema accompanying this standard.

Table B.2— `ReferenceableGridByVectors` structure

Name	Definition	Data type	Multiplicity
`origin`	The origin of the referenceable grid in the external CRS	`GML:: PointPropertyType`	One (mandatory)
`generalGridAxis`	Used to define an offset vector and support parameters	`GeneralGridAxisProperty Type`	One or more (mandatory)

Table B.3— `GeneralGridAxis` structure

Name	Definition	Data type	Multiplicity
`offsetVector`	Specifies a vector in the external CRS	`GML:: VectorType`	One (mandatory)
`coefficients`	Specifies a set of multiplicative coefficients over the grid points	`GML:: doubleList`	One (mandatory)
`gridAxesSpanned`	The names of the grid axes spanned by the coefficients	`GML:: NCNameList`	One (mandatory)
`sequenceRule`	Specifies the order in which the coefficients are applied to the grid points	`GML:: SequenceRuleType`	One (mandatory)

A `generalGridAxis` is followed by a `GeneralGridAxis` that fully specifies an offset vector and its support parameters. The subelement `offsetVector` of `GeneralGridAxis` specifies a single vector in the external CRS. The subelement `coefficients` specifies a corresponding set of coefficients that multiply their respective `offsetVector` at grid points that span one or more of the grid dimensions, which are named with the `gridAxesSpanned` subelement. Finally, the order in which the coefficients are applied over the grid points is indicated using the `sequenceRule` subelement.

`ReferenceableGridByArray` defines a referenceable grid by listing an array of grid point locations explicitly, as a sequence of direct positions in a defined sequence order over the grid.

Requirement 78: https://standards.isotc211.org/19123/-/2/2/req/referenceablegridcoverage/by-array

A `ReferenceableGridByArray` shall be defined by Table B.4 and the XML Schema accompanying this standard.

Table B.4— `ReferenceableGridByArray` structure

Name	Definition	Data type	Multiplicity
`GML::posList` (for example)	Specifies the array of grid point locations in the external CRS, via either a `GML::posList` or a sequence of `GML::pos` or `GML::Point` objects.	`GML:: geometricPosition ListGroup`	One (mandatory)

sequenceRule	Specifies the sequence order of the grid point locations over the grid.	GML::SequenceRuleType	One (mandatory)

GML::Transformation and GML::ConcatenatedOperation specify the relationship between positions in the source CRS and corresponding positions in the target CRS. A sequence of CRSs to be used is optionally defined in gridCRS.

Requirement 79: https://standards.isotc211.org/19123/-/2/2/req/referenceablegridcoverage/by-transformation

A ReferenceableGridByTransformation shall be defined by Table B.5 and the XML Schema accompanying this standard.

Table B.5— ReferenceableGridByTransformation structure

Name	Definition	Data type	Multiplicity
transformation	A general coordinate transformation using a sequence of operations based on GML::method that have an unbounded set of GML::parameterValue	GML::TransformationPropertyType	Zero or one (optional)
concatenatedOperation	An ordered sequence of two or more coordinate operations	GML::ConcatenatedOperationPropertyType	Zero or one (optional)
gridCRS	An optional sequence of CRS definitions used by the transformation or the concatenatedOperation	GridCRSPropertyType	Zero or one (optional)

In a ReferenceableGridBySensorModel the grid transformation of the referenceable grid is defined by a sensor model described with SensorML 2.0 that is used to geolocate the referenceable grid. Such a sensor model involves two inputs: one or more sensor model descriptions containing free variables (using SML::sensorModel) plus a respective set of variable instantiations (using SML::sensorInstance). A sequence of CRS is optionally defined in gridCRS.

NOTE Both sensorModel and sensorInstance are subtypes of SML::AbstractProcessPropertyType that can be followed by instantiable subtypes of SML::AbstractProcess, which include SML::SimpleProcess, SML::AggregateProcess, SML::PhysicalSystem, and SML::PhysicalComponent.

Requirement 80: https://standards.isotc211.org/19123/-/2/2/req/referenceablegridcoverage/by-sensormodel

A ReferenceableGridBySensorModel shall have a structure as given in Table B.6 and the XML schema accompanying this standard.

Table B.6 — ReferenceableGridBySensorModel structure

Name	Definition	Data type	Multiplicity
sensorModel	SensorML model yielding the direct positions of the grid	SML::AbstractProcessPropertyType	One (mandatory)

sensorInstance	Parameter values for the sensor model	SML::AbstractProcessPropertyType	Zero or one (optional)
gridCRS	An optional sequence of CRS definitions used by sensorModel	GridCRSPropertyType	Zero or one (optional)

NOTE 1 If a sensorInstance is specified, it is recommended (following SensorML 2.0 Requirement 13) that its associated SensorML 2.0 document reference its parent sensorModel via a SML::typeOf specification.

NOTE 2 If a sensorInstance is specified, it is recommended that its associated SensorML 2.0 document specify a set of parameter values consistent with the free variables of its parent sensorModel. If a sensorInstance is *not* specified, it is recommended that the parameter values are instead specified within the associated SensorML 2.0 document of the mandatory sensorModel.

NOTE 3 The SensorML 2.0 documents associated with sensorModel and sensorInstance can, following SensorML 2.0 Requirement 40, be specified inline or by reference but cannot be empty.

B.5 Abstract test suite

This clause specifies an abstract test suite which shall be passed in its entirety by any implementation claiming conformance with the coverage types defined in this Annex B.

Reference: All normative statements in class *rectified+referenceablegridcoverage*

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

Reference: All normative statements in class *rectifiedgridcoverage*

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

Reference: All normative statements in class *referenceablegridcoverage*

Test purpose: Verify that the specification under test conforms to all requirements of this conformance class

Test method: Evaluate every requirement in turn; the overall test passes if every single test passes.

Bibliography

[1] IETF RFC 2183, *Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field, Internet Engineering Task Force, 1997*

[2] IETF RFC 2387, The MIME Multipart/Related Content-type, *Internet Engineering Task Force, 1998*

[3] IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax, *Internet Engineering Task Force, 2005*

[4] IETF RFC 2183, Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field, *Internet Engineering Task Force, 1997*

[5] IETF RFC 2392, Content-ID and Message-ID Uniform Resource Locators, *Internet Engineering Task Force, 1998*

[6] IETF RFC 7159, JSON (JavaScript Object Notation), *Internet Engineering Task Force, 2014*

[7] IETF RFC7159, *The* JavaScript Object Notation (JSON) Data Interchange Format. *Internet Engineering Task Force, 2014*

[8] *INSPIRE data specification on elevation*. https://inspire.jrc.ec.europa.eu/documents/Data_Specifications/INSPIRE_DataSpecification_EL_v3.0.pdf

[9] *INSPIRE data specification on orthoimagery*. https://inspire.jrc.ec.europa.eu/documents/Data_Specifications/INSPIRE_DataSpecification_OI_v3.0.pdf

[10] ISO 19123-1:2023, *Geographic information — Schema for coverage geometry and functions — Part 1: Fundamentals*

[11] ISO 19136-1:2020, *Geographic information — Geography Markup Language (GML) — Part 1: Fundamentals*

[12] ISO 19136-2:2015, *Geographic information — Geography Markup Language (GML) — Part 2: Extended schemas and encoding rules*

[13] ISO/IEC 19757-3:2006, *Information technology – Document Schema Definition Languages (DSDL) – Part 3: Rule-based validation – Schematron*

[14] ISO/IEC 19757-3:2016, *Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron*

[15] ISO/TS 19130:2010, *Geographic information – Imagery sensor models for geopositioning.*

[17] OGC 09-110r3, *Web Coverage Service (WCS) Core Interface Standard, version 2*

[18] OGC 09-146r8, *Coverage Implementation Schema, version 1.1.1*

[19] OGC 12-000, *OGC® SensorML: Model and XML Encoding Standard, version 2*

[20] OGC 13-102r2, *Name type specification – Time and index coordinate reference system definitions (OGC Policy Document), version 1*

[21] OGC 14-121, *Web Information Service (WIS), version 1*

[22] OGC 12-000, OGC® SensorML: Model and XML Encoding Standard*, version 2*

[23] OGC 08-131r3, *The ModSpec Model - A Standard for Designing and Writing Modular Standards*.

[24] OGC 16-083, *OGC Implementation Schema for Coverages – ReferenceableGridCoverage Extension*.

[25] OGC 09-110r4, *Web Coverage Service (WCS) Core Interface Standard, version 2.0.1*

[26] OGC 09-146r2, *Coverage Implementation Schema, version 1.0.1*

[27] OGC 12-100r1, *OGC® GML Application Schema — Coverages — GeoTIFF Coverage Encoding Profile, version 1.0.1*

[29] OGC 14-100r2, *CF-NetCDF 3.0 encoding using GML Coverage Application Schema version 2.0, version 2.0.0*

[30] UNIFIED CODE FOR UNITS OF MEASURE (UCUM). https://unitsofmeasure.org

[31] W3C JSON-LD 1.0 Processing Algorithms and API. https://www.w3.org/TR/json-ld-api

[32] W3C JSON-LD 1.0, A JSON-based Serialization for Linked Data. https://www.w3.org/TR/json-ld/

[33] W3C Recommendation, *XML Linking Language (XLink)*, version 1, 2001

[34] W3C Recommendation, *XML Path Language (XPath)*, version 2, 2007

[35] W3C Recommendation, *XML Path Language (XPath)*, version 2.0.1, 2007 (www.w3.org/xpath20)

[36] W3C Working Draft, *The app: URI scheme*, 2013

[37] OGC 16-083r3, OGC Coverage Implementation Schema - ReferenceableGridCoverage Extension with Corrigendum*, version 1.0.1*

[38] OGC 10-129r1, OGC Geographic Markup Language (GML) — Extended schemas and encoding rules, *version 3.3*

[39] OGC 08-068r3, OGC Web Coverage Processing Service *(*WCPS*), version 1.1*

[40] ISO 19103:2024, *Geographic information — Conceptual schema language*

[41] W3C, *CURIE Syntax 1.0*, https://www.w3.org/TR/curie/

[42] ISO 19123-3:2023, *Geographic information — Schema for coverage geometry and functions — Part 3: Processing fundamentals*

[43] Regenstrief Institute: *The Unified Code for Units of Measure*. https://ucum.org/ucum