1 Implementation: Basic Grid Elements

GeoAPI package:	org.opengis.coverage.qgrid
GeoTools package:	org.geotools.coverage.qgrid
Implementor:	*Your Name Here (change text style to "Table Contents" when entered.)*
Specified Implementation Classes:	● Grid ● GridEnvelope ● GridValuesMatrix ● GridPositioning ● RectifiedGrid ● SequenceRule ● ByteGridCoordinates3D ● ByteGridCoordinates4D ● ShortGridCoordinates2D ● ShortGridCoordinates3D ● ShortGridCoordinates4D ● IntGridCoordinates2D ● ByteGridCoordinatesFactory ● ShortGridCoordinatesFactory ● IntGridCoordinatesFactory
Specified Interface Classes:	● Grid ● GridEnvelope ● GridValuesMatrix ● GridPositioning ● RectifiedGrid ● ReferencableGrid ● SequenceRule ● SequenceType ● GridCell ● GridPoint ● FootPrint ● GridCoordinates ● GridCoordinatesFactory

1.1 Summary

The Basic Grid Elements implementation unit concerns the core of grid-related classes. These classes are the foundation on which gridded coverages are built. Only three classes in this implementation unit require a coordinate reference system: FootPrint, RectifiedGrid, and ReferencableGrid. ReferencableGrid is not implemented here, as it must be implemented by a user with highly specialized knowledge about the grid in question.

A significant fraction of the classes rightfully belonging to this section are left unimplemented. This is due to the efficiency concerns raised in chapter 5 of the ISO 19123 Primer. Primarily, the omitted classes are the ones which must be instantiated for every grid point or grid cell. Prior to implementation, some means must be found to strictly control the creation of instances (e.g., tiling, object pooling, lazy object creation, etc.) or they must be made more memory efficient, or both. Implementation of these classes is deferred until after a review and discussion by the wider GeoTools community. Interfaces, however, are specified, as they are unlikely to change.

1.2 Departure from ISO 19123

There are two structural departures from the classes specified in ISO 19123. One departure is to avoid multiple inheritance specified by the standard, and the other is to fix a type issue. A non-structural change is the addition of a GridCoordinateFactory interface for the creation of GridCoordinates and GridEnvelopes.

1.2.1 Avoiding Multiple Inheritance

The details of this problem, as well as the proposed solution, may be found in the section “Indexed access to a CV_Grid” in the ISO 19123 Primer.

The single largest change to the basic grid types outlined in ISO 19123 is embodied in Figure 1. The items in the green box have been rearranged to avoid the multiple inheritance specified by ISO 19123. The standard specifies that a particular instance of CV_GridValuesMatrix may also be an instance of either CV_ReferencableGrid or CV_RectifiedGrid. The reformulation in Figure 1 specifies this same arrangement with the use of composition instead of inheritance. The new associations, Positioning and Valuation, have been made containment relationships because ISO19123 specifies them as inheritance relationships and it was desirable to retain this tight coupling.

A second, minor change, is that the EvaluationStructure association has been made optional. This is due to the fact that the arrangement of CV_GridPoints may not be such that the construction of a CV_GridCell is possible (e.g., they could all be in a single row or column.)

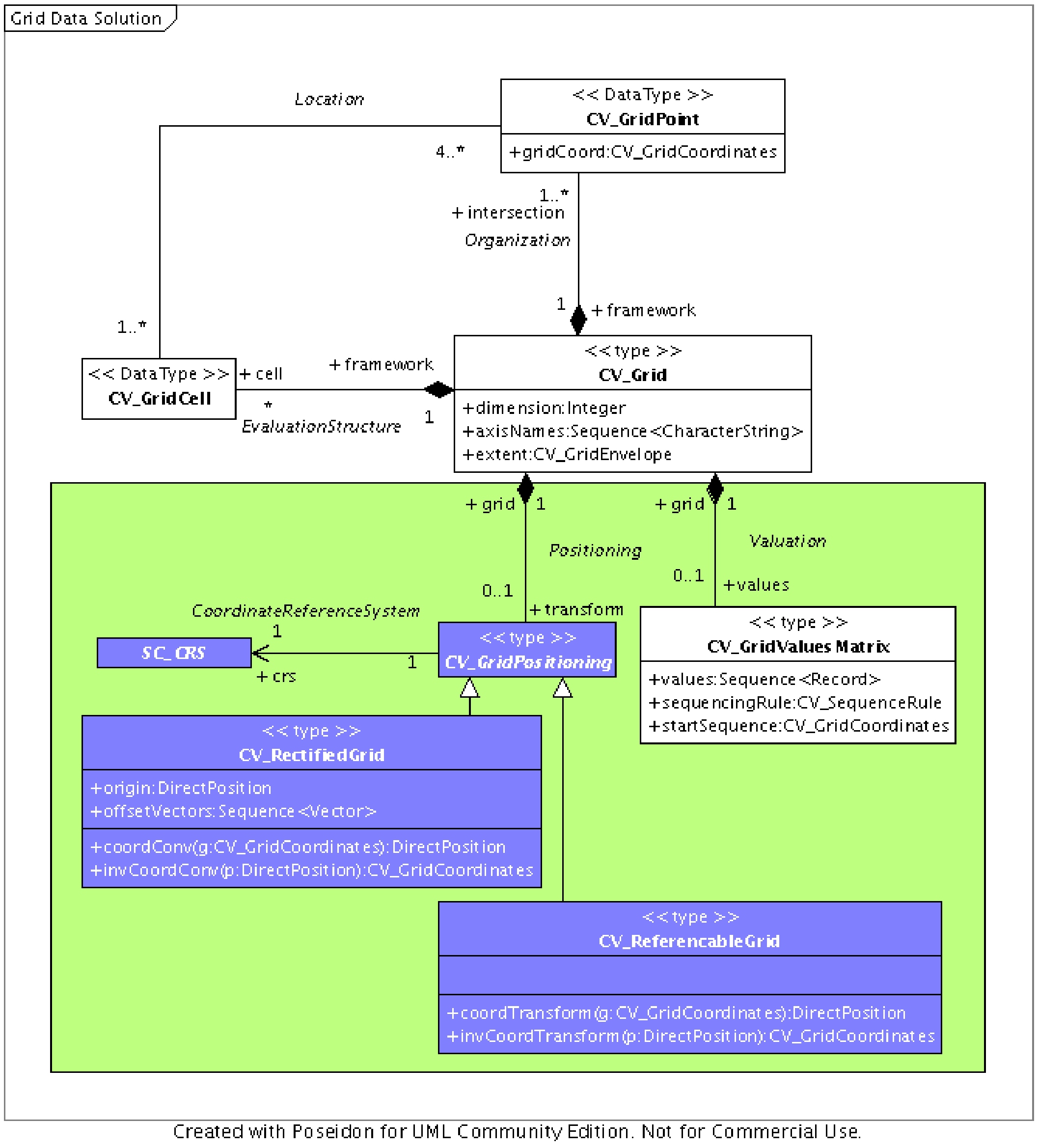
Grid Data Solution
Location
<< DataType >>
CV_GridPoint
+gridCoord:CV_GridCoordinates
4..*
1..*
+intersection
Organization
1
+framework
1..*
<< DataType >>
CV_GridCell
+cell
+framework
*
EvaluationStructure
1
<< type >>
CV_Grid
+dimension:Integer
+axisNames:Sequence<CharacterString>
+extent:CV_GridEnvelope
+grid
1
+grid
1
Positioning
Valuation
0..1
0..1
+values
+transform
CoordinateReferenceSystem
1
SC_CRS
+crs
1
<< type >>
CV_GridPositioning
<< type >>
CV_GridValuesMatrix
+values:Sequence<Record>
+sequencingRule:CV_SequenceRule
+startSequence:CV_GridCoordinates
<< type >>
CV_RectifiedGrid
+origin:DirectPosition
+offsetVectors:Sequence<Vector>
+coordConv(g:CV_GridCoordinates):DirectPosition
+invCoordConv(p:DirectPosition):CV_GridCoordinates
<< type >>
CV_ReferencableGrid
+coordTransform(g:CV_GridCoordinates):DirectPosition
+invCoordTransform(p:DirectPosition):CV_GridCoordinates
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figure 1: Grid data types: Reworked to avoid multiple inheritance.

1.2.2 Parentage of CV_GridCell

The parentage of class CV_GridCell is dealt with in the "Specialized Value Objects" IWUG. While the justification for changing the parentage is presented there, the specification of the class is presented in this IWUG, as it is intimately coupled with other objects in this IWUG. In essence, CV_GridCell must inherit from CV_DomainObject in order to be consistent with the usage of other specializations of CV_ValueObject.

1.2.3 Availability of a MathTransform for RectifiedGrids

ISO 19123 does not provide a way to convert from a location in an external CRS to a non-integer grid coordinate. This implementation does supply such a transformation in the case of a RectifiedGrid, which is guaranteed to have an affine relationship from grid indices to external coordinates. A ReferenceableGrid may encapsulate a much more complex relationship and therefore does not include this attribute.

1.3 GeoAPI Interface Diagrams

1.3.1 Grid Coordinate Interfaces

The interfaces shown in Figure 2 depict the Grid Coordinate interfaces defined in GeoAPI. Note the addition of the GridCoordinateFactory and the fact that it creates both GridCoordinates and GridEnvelopes.

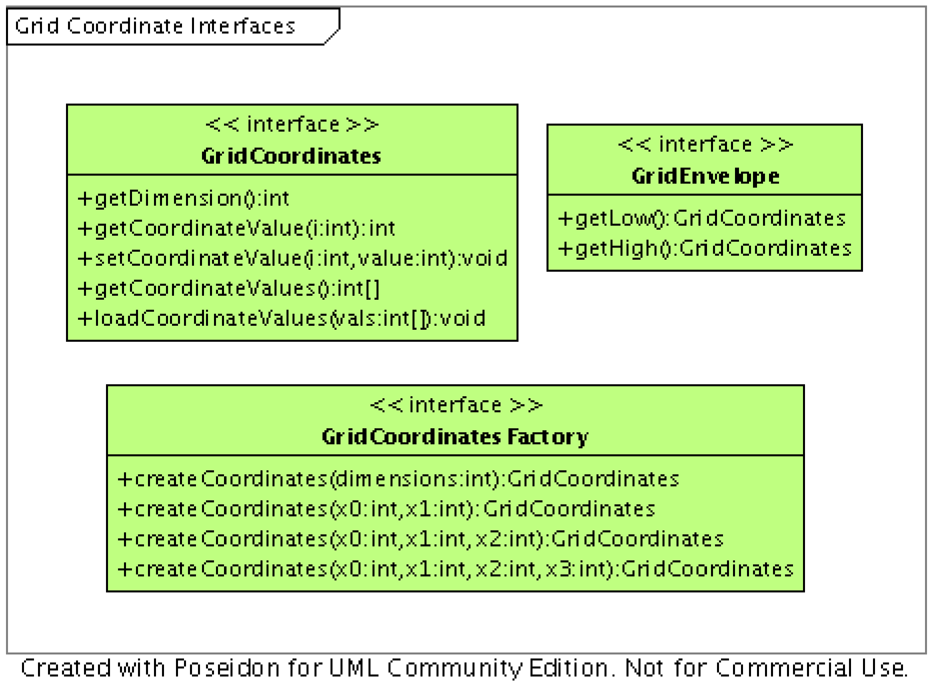
Grid Coordinate Interfaces
<< interface >>
GridCoordinates
+getDimension():int
+getCoordinateValue(i:int):int
+setCoordinateValue(i:int,value:int):void
+getCoordinateValues():int[]
+loadCoordinateValues(vals:int[]):void
<< interface >>
GridEnvelope
+getLow():GridCoordinates
+getHigh():GridCoordinates
<< interface >>
GridCoordinatesFactory
+createCoordinates(dimensions:int):GridCoordinates
+createCoordinates(x0:int,x1:int):GridCoordinates
+createCoordinates(x0:int,x1:int,x2:int):GridCoordinates
+createCoordinates(x0:int,x1:int,x2:int,x3:int):GridCoordinates
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figure 2: GeoAPI Grid Coordinate Interfaces for Basic Grid Elements.

1.3.2 Gridded Data Interfaces

The interfaces shown in Figure 3 provide access to gridded data. These classes do not represent Coverage classes, but they are tools with which gridded coverages may be built. Related classes are collected within blue boxes. Note that the departure from ISO 19123 suggested in Figure 1 is represented by the interfaces in Figure 3. The Grid interface has no children. The Valuation and Positioning partitions specified in ISO 19123 are implemented by composition instead of inheritance. The GridPositioning interface is shown as abstract because implementations should never make a GridPositioning interface instantiable.

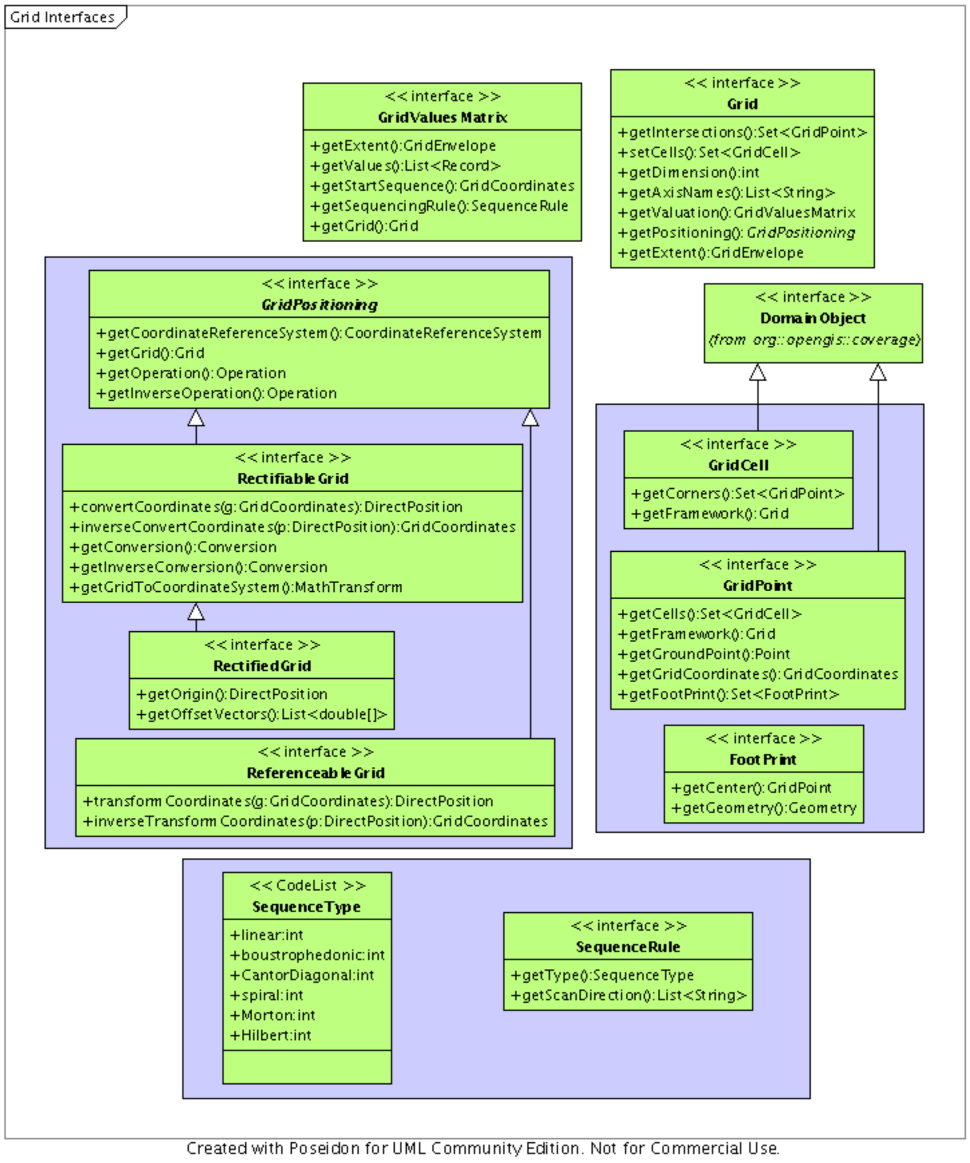
Grid Interfaces
<< interface >>
GridValuesMatrix
+getExtent():GridEnvelope
+getValues():List<Record>
+getStartSequence():GridCoordinates
+getSequencingRule():SequenceRule
+getGrid():Grid
<< interface >>
Grid
+getIntersections():Set<GridPoint>
+setCells():Set<GridCell>
+getDimension():int
+getAxisNames():List<String>
+getValuation():GridValuesMatrix
+getPositioning():GridPositioning
+getExtent():GridEnvelope
<< interface >>
GridPositioning
+getCoordinateReferenceSystem():CoordinateReferenceSystem
+getGrid():Grid
+getOperation():Operation
+getInverseOperation():Operation
<< interface >>
DomainObject
(from org::opengis::coverage)
<< interface >>
RectifiableGrid
+convertCoordinates(g:GridCoordinates):DirectPosition
+inverseConvertCoordinates(p:DirectPosition):GridCoordinates
+getConversion():Conversion
+getInverseConversion():Conversion
+getGridToCoordinateSystem():MathTransform
<< interface >>
GridCell
+getCorners():Set<GridPoint>
+getFramework():Grid
<< interface >>
GridPoint
+getCells():Set<GridCell>
+getFramework():Grid
+getGroundPoint():Point
+getGridCoordinates():GridCoordinates
+getFootPrint():Set<FootPrint>
<< interface >>
RectifiedGrid
+getOrigin():DirectPosition
+getOffsetVectors():List<double[]>
<< interface >>
FootPrint
+getCenter():GridPoint
+getGeometry():Geometry
<< interface >>
ReferenceableGrid
+transformCoordinates(g:GridCoordinates):DirectPosition
+inverseTransformCoordinates(p:DirectPosition):GridCoordinates
<< CodeList >>
SequenceType
+linear:int
+boustrophedonic:int
+CantorDiagonal:int
+spiral:int
+Morton:int
+Hilbert:int
<< interface >>
SequenceRule
+getType():SequenceType
+getScanDirection():List<String>
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figure 3: GeoAPI interfaces for gridded data access.

1.4 Implementation Design

1.4.1 GridCoordinates Implementation

The specified implementations of the GridCoordinates interface are given in Figure 4. These implementations are backed by byte, short, and int primitive numeric types in Java, as suggested in Chapter 5 of the ISO 19123 Primer. An integer implementation is supplied only for the 2D case. Short integer implementations are supplied for 2, 3, and 4 dimensional cases. Byte integer implementations are supplied only for the three and four dimensional cases. An AbstractGridCoordinates class is specified to provide an implementation of the loadCoordinateValues method to be inherited by all the children.

Each set of GridCoordinate implementation classes has its own coordinate factory: one for byte, one for short, and one for integer data. Again, an AbstractGridCoordinatesFactory is specified to provide a default implementation for the various createCoordinates() methods. This default implementation is to throw IndexOutOfBoundsException. Children then override only the methods they support.

Lastly, the implementation of GridEnvelope is a straightforward class with two attributes of type GridCoordinate.

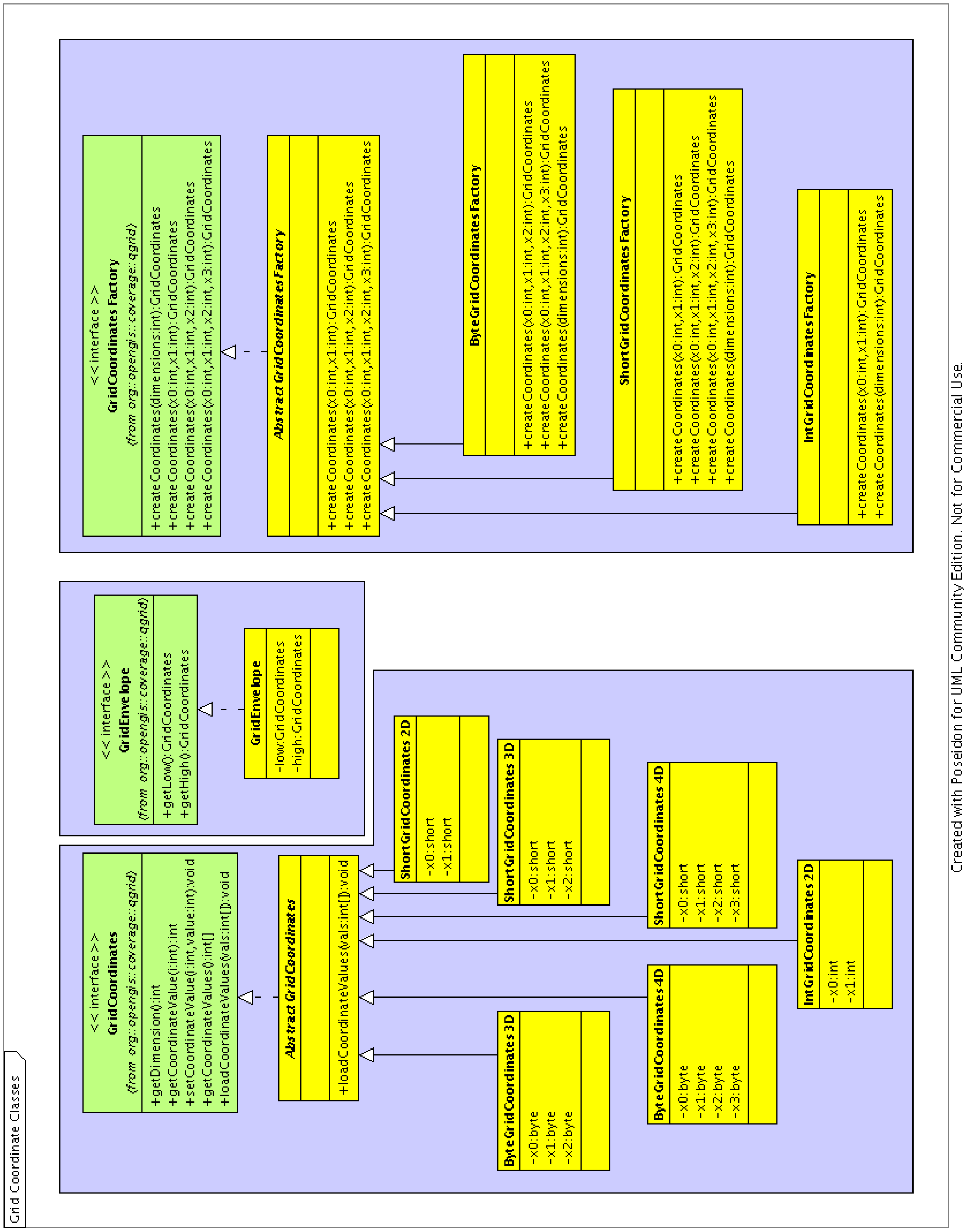
Grid Coordinate Classes
<<interface>>
GridCoordinates
(from org::opengis::coverage::ggrid)
+getDimension():int
+getCoordinateValue(i:int):int
+setCoordinateValue(i:int,value:int):void
+getCoordinateValues():int[]
+loadCoordinateValues(vals:int[]):void
AbstractGridCoordinates
+loadCoordinateValues(vals:int[]):void
ByteGridCoordinates3D
-x0:byte
-x1:byte
-x2:byte
ByteGridCoordinates4D
-x0:byte
-x1:byte
-x2:byte
-x3:byte
ShortGridCoordinates2D
-x0:short
-x1:short
ShortGridCoordinates3D
-x0:short
-x1:short
-x2:short
ShortGridCoordinates4D
-x0:short
-x1:short
-x2:short
-x3:short
IntGridCoordinates2D
-x0:int
-x1:int
<<interface>>
GridEnvelope
(from org::opengis::coverage::ggrid)
+getLow():GridCoordinates
+getHigh():GridCoordinates
GridEnvelope
-low:GridCoordinates
-high:GridCoordinates
<<interface>>
GridCoordinatesFactory
(from org::opengis::coverage::ggrid)
+createCoordinates(dimensions:int):GridCoordinates
+createCoordinates(x0:int,x1:int):GridCoordinates
+createCoordinates(x0:int,x1:int,x2:int):GridCoordinates
+createCoordinates(x0:int,x1:int,x2:int,x3:int):GridCoordinates
AbstractGridCoordinatesFactory
+createCoordinates(x0:int,x1:int):GridCoordinates
+createCoordinates(x0:int,x1:int,x2:int):GridCoordinates
+createCoordinates(x0:int,x1:int,x2:int,x3:int):GridCoordinates
ByteGridCoordinatesFactory
+createCoordinates(x0:int,x1:int,x2:int):GridCoordinates
+createCoordinates(x0:int,x1:int,x2:int,x3:int):GridCoordinates
+createCoordinates(dimensions:int):GridCoordinates
ShortGridCoordinatesFactory
+createCoordinates(x0:int,x1:int):GridCoordinates
+createCoordinates(x0:int,x1:int,x2:int):GridCoordinates
+createCoordinates(x0:int,x1:int,x2:int,x3:int):GridCoordinates
+createCoordinates(dimensions:int):GridCoordinates
IntGridCoordinatesFactory
+createCoordinates(x0:int,x1:int):GridCoordinates
+createCoordinates(dimensions:int):GridCoordinates
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figure 4: GridCoordinates implementation design for Basic Grid Elements.

1.4.2 Gridded Data Implementation

Classes shown in Figure 5 show the implementation required to provide basic gridded data access functionality.

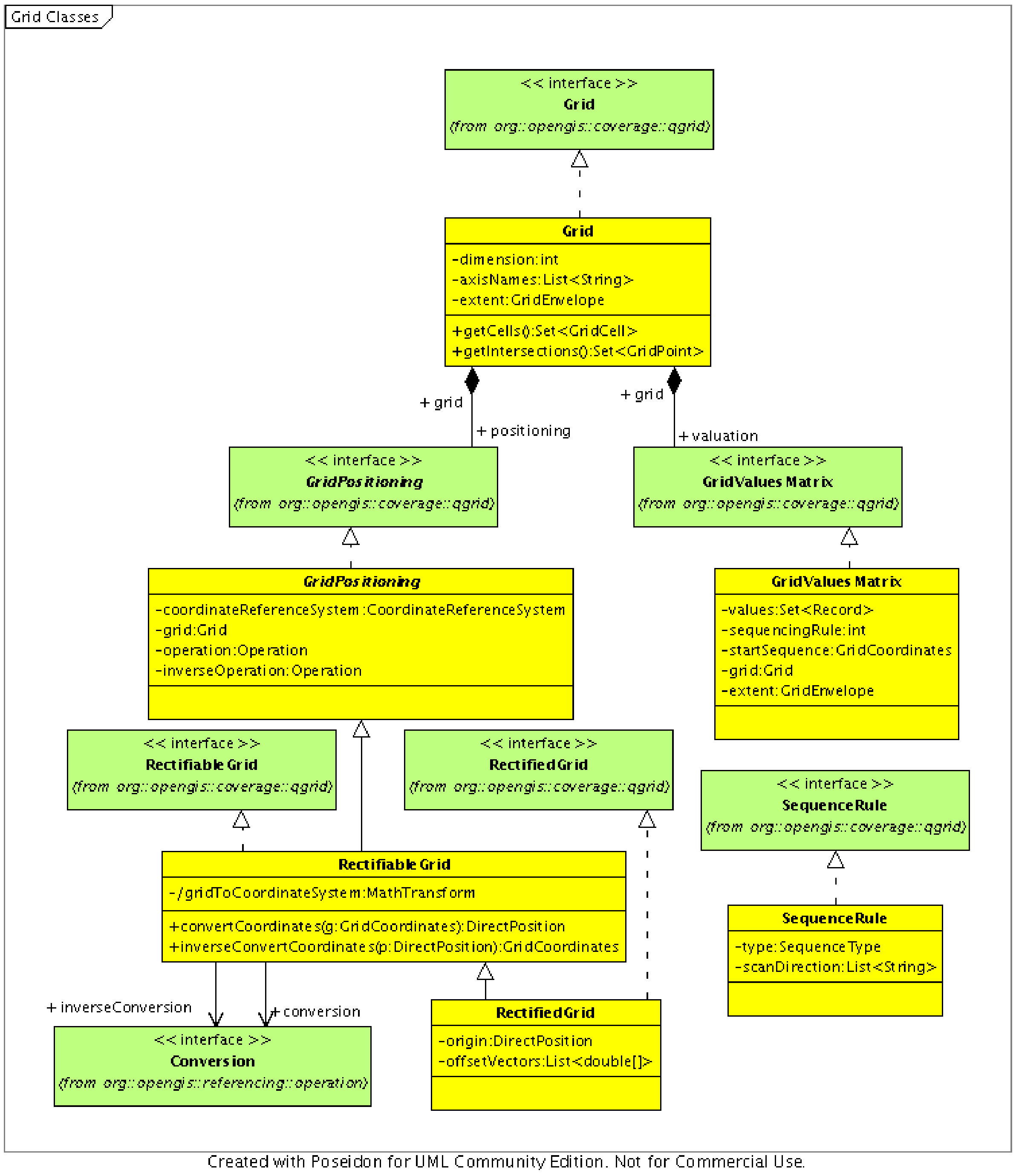
Grid Classes
<< interface >>
Grid
(from org::opengis::coverage::qgrid)
Grid
-dimension:int
-axisNames:List<String>
-extent:GridEnvelope
+getCells():Set<GridCell>
+getIntersections():Set<GridPoint>
+grid
+positioning
+grid
+valuation
<< interface >>
GridPositioning
(from org::opengis::coverage::qgrid)
<< interface >>
GridValuesMatrix
(from org::opengis::coverage::qgrid)
GridPositioning
-coordinateReferenceSystem:CoordinateReferenceSystem
-grid:Grid
-operation:Operation
-inverseOperation:Operation
GridValuesMatrix
-values:Set<Record>
-sequencingRule:int
-startSequence:GridCoordinates
-grid:Grid
-extent:GridEnvelope
<< interface >>
RectifiableGrid
(from org::opengis::coverage::qgrid)
<< interface >>
RectifiedGrid
(from org::opengis::coverage::qgrid)
<< interface >>
SequenceRule
(from org::opengis::coverage::qgrid)
RectifiableGrid
-/gridToCoordinateSystem:MathTransform
+convertCoordinates(g:GridCoordinates):DirectPosition
+inverseConvertCoordinates(p:DirectPosition):GridCoordinates
SequenceRule
-type:SequenceType
-scanDirection:List<String>
+inverseConversion
+conversion
<< interface >>
Conversion
(from org::opengis::referencing::operation)
RectifiedGrid
-origin:DirectPosition
-offsetVectors:List<double[]>
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figure 5: Implementation of types which provide basic gridded data access.

1.5 Detailed Interface-only Discussion

Interfaces in this section are intended to be implemented only as interfaces in GeoAPI.

1.5.1 GridCell

Identifier:	CV_GridCell

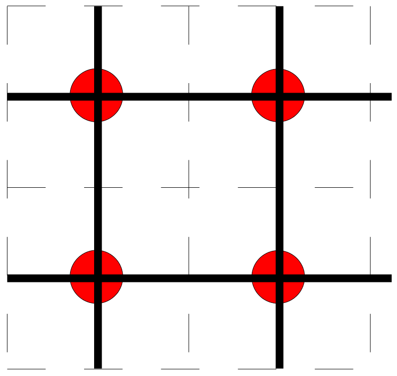

Figure 6: Illustration of Quadrilateral Grid concepts.

The definition of a GridCell and related classes may be found in the Quadrilateral Grid Cell Concepts chapter of the ISO 19123 Primer. Succinctly, a grid cell is the area inside the four grid points which act as corners (in 2D). Copied here for convenience is Figure 6, which illustrates the relationship of a GridCell to the GridPoints which act as corners. This illustration applies only to the 2D case and it is left to the reader to extrapolate this drawing to higher dimensions. As shown, the red dots represent grid points, the heavy black lines represent “grid lines” (a concept for which no class or interface is defined), and the area bounded on all sides by grid lines is the grid cell.

Note that the grid cell is not associated with any single grid point, but rather is associated equally with all the corner points.

The function of the grid cell data structure is identical to the conceptual function of the grid cell definition. It merely provides a vehicle with which the corner points may be associated.

Note that this data structure is ripe for optimization. As defined, the grid cell is a fundamental unit of the interior of a grid. In other words, the grid points forming the corners are adjacent to one another. Along any axis, there are no grid points between two corner points. As such, a grid cell may be uniquely defined by storing only one corner and generating the rest on demand using a set of rules. If the stored grid point is defined to be the corner nearest to the origin, then generating the remaining corners becomes easy.

A grid cell is defined entirely by the coordinate system of the grid. A Grid Cell is not represented in an external coordinate reference system.

1.5.1.1 corners

Identifier:	corners

The corners association contains a set of grid points which define the bounding area (2D), volume (3D), or hyper-volume (4D+) of this grid cell. The Set returned by the accessor method should be optimized to store only the data necessary to generate the remaining corners if required.

1.5.1.2 framework

Identifier:	framework

The framework role associates this grid cell with the grid to which it belongs. All effort should be made to implement this accessor method such that it does not rely on stored data to return a value.

1.5.1.3 spatialElements

Identifier:	Extension

This role name is inherited from DomainObject and associates the grid cell with a geometric object which encodes only the spatial components of the grid index. Spatial axes in the geometric object are specified in the same order as in the grid coordinates. The temporal axis and any categorical axes are omitted. In two spatial dimensions, the geometric object shall be a GM_PolyhedralSurface composed of a single GM_Polygon. For three spatial dimensions, the user must specify a GM_Solid object which represents the volume bounded by the eight corners.

Because we consider time to be orthogonal to space, these spatial elements may be factored out. The same spatial elements participate at the start time and at the end time. Therefore, they need only be represented once.

1.5.1.4 temporalElements

Identifier:	Extension

This role name is inherited from DomainObject and associates the grid cell with a TM_Interval which represents the two TM_Instants which participate in the Grid Cell. Because there may be only one time axis, there can be only two relevant TM_Instants.

Because we consider time to be orthogonal to space, these temporal elements may be factored out and represented separately from the spatial elements.

1.5.2 GridPoint

Identifier:	CV_GridPoint

A grid point exists at every combination of integer coordinate values within the domain of the coverage. Grid points do not have any spatial extent and serve only as an index into the gridded values. As shown in Figure 6, grid points of a 2D grid may serve as a corner for up to four grid cells.

A grid point is conceptually associated with a sample point and a sample space. These are physically meaningful concepts which are not represented explicitly (with classes) in ISO 19123. A sample point is the location of a sample in the grid coordinate system. A sample space is the area (again in the grid coordinate system) which participates in the measurement (e.g., the field of view of a pointing radiometer, or the instantaneous field of view of a scanning detector.) The grid point itself, however, is neither of these concepts: it is merely an index.

ISO 19123 does explicitly specify a “footprint”, which is the representation of the sample space in some external coordinate system. In the specification, this footprint is associated with the grid cell via the *SampleSpace* relation. Associating the grid point and the footprint in this way specifies the role played by the sample space concept: it is a transparent concept which produces a sampling artifact (the footprint) on the ground. The association itself embodies specialized knowledge about the measurement process which is capable of calculating the area (in an external coordinate system), which participated in the measurement identified by the grid point index. Multiple footprints may be associated with a single grid point. This facility allows for the representation of the sample space in many different external coordinate systems.

1.5.2.1 gridCoordinates

Identifier:	gridCoord

Integer representation of the index into the grid. The index has no units, all axes are orthogonal (in grid space), and no distinction is made between spatial and temporal axes. The grid coordinates must occur in the same order as the axes names in the grid associated to this point by “framework”.

1.5.2.2 cells

Identifier:	cell

Associates this point with the cells for which it acts as a corner. In 2D, this point may be a corner for up to four cells.

1.5.2.3 framework

Identifier:	framework

Associates this point with the grid for which it represents an index.

1.5.2.4 groundPoint

Identifier:	groundPoint
Obligation:	Optional

Represents the location of the grid point (not the sample point) in an external coordinate reference system. This distinction between grid point and sample point is important. Remember: the grid point is just an index and nothing more. The associated sample point is the quantity related to the measurement. Therefore, the ground point is a representation of the grid index on the ground, and should not be taken to represent the location of the measurement.

1.5.2.5 footPrint

Identifier:	footprint
Obligation:	Optional

Represents the sample space in one or more external coordinate reference systems.

1.5.3 FootPrint

The FootPrint represents the area which participated in the measurement. Indeed, as the FootPrint is not constrained to two dimensional geometry, it is entirely possible for volumetric “areas” to be represented (e.g., the sample volume of a radar or lidar, the entire column of a column water vapor measurement, or the light-path between a ground based sunphotometer and the sun).

1.5.3.1 geometry

Identifier:	geometry

Contains the actual area (or length, or volume) which produced the measurement with which this footprint is associated via the “center” relation.

1.5.3.2 index

Identifier:	center

This role name in ISO 19123, “center”, is a misnomer. By definition, the sample point represented in the same external coordinate reference system as the footprint, is the “center” of the footprint. This “center” attribute associates the footprint with a grid point, which represents only the index into the grid. This implementation specification, therefore, has renamed the association role to “index”.

1.5.4 ReferenceableGrid

Identifier:	CV_ReferenceableGrid

A ReferencableGrid is a gridded set of data values with a uniquely determined but non-simple transformation into an external CRS. If the grid has a simple affine transformation into an external CRS, see the definition of a RectifiedGrid in section 1.6.8. A ReferenceableGrid implements a *coordinate transformation* operation, as specified by ISO 19111, which means that any parameters used are empirically derived and the operation involves a datum change.

Examples of a referenceable grid would be uncorrected aerial photography or satellite imagery. Additionally, any situation where individual coordinates are stored with (or calculated for) each pixel could be considered referencable.

ISO 19123 does not specify what quantity shall be transformed by this class. The GridCoordinates which participate in the methods are the indices into the grid. The sample point may be offset from this. This class may therefore transform the actual grid coordinates provided, or it may transform the associated sample point. Each implementation shall state clearly what action is being performed.

A mechanism to transform coordinates from an external CRS to non-integer coordinates in the grid's EngineerningCRS has intentionally been left out of this class. In the case of unrectified satellite imagery, the same location on the ground may be represented by different, noncontiguous pixels (e.g., “bowtie effect”.) Such a mechanism applies to a RectifiedGrid.

1.5.4.1 coordinateReferenceSystem

Identifier:	crs

This inherited association from GridPositioning specifies the coordinate system into which this object transforms coordinates. ISO 19123 specifies this association directly on this class.

1.5.4.2 transformCoordinates

Identifier:	coordTransform

Transforms the specified GridCoordinates to a location in an external CRS. This method may transform the GridCoordinates directly, or may transform the associated sample point.

1.5.4.3 inverseTransformCoordinates

Identifier:	invCoordTransform

Transforms the specified coordinate in an external CRS to the GridCoordinate which represents this location. If there is more than one GridCoordinate which represents the specified location, this method returns the GridCoordinate associated with the closest sample point.

1.5.4.4 inverseTransformCoordinateList

Identifier:	Extension
Obligation:	Optional

Transforms the specified coordinate in an external CRS to the list of GridCoordinates which represents this location. The list is ordered by increasing distance from specified location to the location of associated sample points on the ground.

1.5.5 SequenceType

Identifier:	CV_SequenceType

This is a CodeList to specify how the multidimensional grid of values is mapped to the one-dimensional List of Records in the Grid. For a complete description of each of the predefined algorithms, see appendix D of ISO 19123.

1.6 Detailed Interface-and-Implementation Discussion

Items discussed in this section require implementation as GeoAPI interfaces and as GeoTools classes. In some cases (most notably the case of GridCoordinates) several GeoTools implementations are provided for a single GeoAPI interface.

1.6.1 GridCoordinates

Identifier:	CV_GridCoordinates

Grid coordinates are the integer indices into the grid. This class is ripe for optimization, as every GridPoint contains one set of grid coordinates. As such, this implementation specifies GridCoordinates implementations which are backed by byte, short, and integer data types. Different dimensions are explicitly hardcoded as the representation of coordinate data in fields, as this is more efficient than using arrays and the number of common use cases is well constrained. The larger numeric data types are not defined for the higher dimensions, as it is unlikely that the increased dynamic range will be required. Users should use the smallest implementation which suits their requirements.

1.6.1.1 dimension

Identifier:	Extension

Returns the number of dimensions represented by this implementation of GridCoordinates. This value may be hardcoded.

1.6.1.2 coordinateValue

Identifier:	Extension

Allows the user to set or retrieve individual coordinate values. Both users and implementors should consider these accessors to be the preferred means of accessing data and should treat these methods accordingly. Users should use these accessor methods whenever possible. Implementors should make these methods as efficient as possible.

1.6.1.3 coordinateValues

Identifier:	coordValues

This accessor represents the coordinate values as an array of integers. This should return a copy of the data and not an array which backs the GridCoordinates implementation object. Users should prefer to use the coordinateValue accessor methods (which provides access to individual elements) over this method.

1.6.1.4 loadCoordinateValues

Identifier:	Extension

This is a compromise method which loads the values of this GridCoordinates implementation into the array provided by the user. Use of this method should be encouraged by those desiring to access the grid coordinates as an array.

1.6.2 GridEnvelope

Identifier:	CV_GridEnvelope

This data type specifies a region of the grid that is bounded by low and high values along each axis.

1.6.2.1 low

Identifier:	low

Specifies the lower bound of each axis.

1.6.2.2 high

Identifier:	high

Specifies the upper bound of each axis.

1.6.3 GridCoordinatesFactory

Identifier:	Extension

This is the primary method of constructing GridCoordinates. Each GridCoordinatesFactory is associated with a single backing data type (e.g., byte, short, or integer), and is capable of manufacturing GridCoordinates of a specified dimensionality. Specialized methods are provided to create GridCoordinates objects which are initialized to the supplied two, three or four-dimensional values.

1.6.3.1 createCoordinates

Identifier:	Extension

There are four variants on this method. The first variant allows the user to specify the dimensionality of the desired GridCoordinates object, but does not specify the initial values. This will create an uninitialized object of the desired dimensionality if this factory is capable of it. The other variants supply only the desired initial values for the grid coordinates object and the dimensionality is derived by the number of parameters supplied.

If the factory is unable to create an object of the desired dimensionality, this method should throw an IndexOutOfBoundsException.

1.6.4 Grid

Identifier:	CV_Grid

Contains the geometric characteristics of a qualdrilateral grid. A grid is a network composed of two or more sets of curves in which members of each set intersect the members of other sets in a systematic way. The curves are called *grid lines*; the points at which they intersect are *grid points*; the interstices between the grid lines are called *grid cells.* More comprehensive documentation may be found in the ISO 19123 Primer.

A grid has two primary composition associations, both of which are optional. The first, *Valuation*, provides values for every grid point. The second, *Positioning*, is capable of transforming the grid point (or the associated sample point) to an associated point in an external coordinate system. The positioning association may link the Grid with one of two types of Positioning object: a rectified grid or a referenceable grid. A rectified grid has a simple, affine relationship from the grid points to locations in the external coordinate system. A referenceable grid object represents an operation of arbitrary complexity. This could be as simple as projecting the coordinates to a new coordinate system or as complex as correcting sampling effects in imagery (e.g. bowtie effects in MODIS imagery.)

A grid with neither composition defined provides only the ability to represent the domain of the grid.

1.6.4.1 dimension

Identifier:	dimension

Returns the dimensionality of the grid. The dimensionality is the number of definining curve sets that constitute the grid.

1.6.4.2 axisNames

Identifier:	axisNames

Returns a list containing the names of the grid axes. Each name is linked to one of the defining curve sets that constitute the grid. Users should observe the naming constraints in Table 18 of ISO 19111 when assigning names to axes. In addition to the constraints in ISO 19111, the name of any temporal coordinate system should be “time”. These restrictions are collected in Table 1 for reference. To place this in perspective, ISO 19111 specifically places two restrictions on the names in Table 1: if the axis is described by CS and CRS columns associated with a name in the table, then the appropriate name shall be used; and a name in Table 1 shall not be used to describe any CS/CRS combination other than the one listed.

CS	***CRS***	***Permitted coordinate system axis names***
Cartesian	geocentric	geocentric X, geocentric Y, geocentric Z
spherical	geocentric	spherical latitude, spherical longitude, geocentric radius
ellipsoidal	geographic	geodetic latitude, geodetic longitude, ellipsoidal height (if 3D)
vertical	vertical	depth, gravity-related height
Cartesian	projected	northing, southing, easting, westing
temporal	temporal	time

Table 1: Summary of naming constraints from ISO 19111 + temporal axis constraint.

1.6.4.3 extent

Identifier:	extent

Returns the limits of a section of the grid. The envelope contains the low and high coordinates of the minimal envelope that can contain the grid. If the Valuation composition is defined, this extent parameter must accurately represent the window for which the associated GridValuesMatrix object provides values.

1.6.4.4 cells

Identifier:	cell
Obligation (ISO19123):	Mandatory
Obligation (GeoAPI):	Optional

Returns the set of grid cells delineated by the grid lines. The collection contains one or more grid cells. This set shall dynamically generate the domain representation and shall not, under any circumstances, cause all members of the Set to be created in memory at once.

This association is optional because a grid is only required to have one grid point, from which it is impossible to define a cell. ISO 19123 erroneously specifies this association as mandatory while failing to guarantee that sufficient data exists to construct a grid cell.

The implementation of this association is deferred until a future date, when efficiency issues have been more thoroughly considered. As of now, attempting to retrieve this attribute should throw an UnsupportedOperationException.

1.6.4.5 intersections

Identifier:	intersection

Returns the set of grid points that are located at the intersections of the grid lines. The collection contains one or more grid points. This set shall dynamically generate the domain representation and shall not, under any circumstances, cause all members of the Set to be created in memory at once.

The implementation of this association is deferred until a future date, when efficiency issues have been more thoroughly considered. As of now, attempting to retrieve this attribute should throw an UnsupportedOperationException.

1.6.4.6 positioning

Identifier:	Extension

Specified in ISO 19123 as a “partition” of an inheritance relation, the positioning facility is recast here as a composition association. This increases clarity and eliminates the required multiple inheritance. The positioning association shall link this grid with an object capable of transforming the grid coordinates into a representation in an external coordinate reference system. The associated object may be either a RectifiedGrid or a ReferenceableGrid, but shall not be only a GridPositioning object.

1.6.4.7 valuation

Identifier:	Extension

Specified in ISO 19123 as a “partition” of an inheritance relation, the valuation facility is recast here as a composition association. This increases clarity and eliminates the required multiple inheritance. The valuation association organizes the multi-dimensional grid into a linear sequence of values according to a limited number of specifiable schemes.

1.6.5 GridValuesMatrix

Identifier:	CV_GridValuesMatrix

Ties feature attributes values to the grid geometry. This class represents an encapsulation of mapping an n-dimensional grid of Records onto a one-dimensional Sequence, using one of a number of predefined algorithms.

1.6.5.1 values

Identifier:	values

This list of Records is the representation of the feature attribute values in the window specified by the “extent” parameter. The order of Records in the list is defined by the algorithm specified in the sequencingRule attribute, starting with the coordinate specified in the startSequence atrtibute.

1.6.5.2 sequencingRule

Identifier:	sequencingRule

Describes how the grid points are ordered. This parameter specifies both the algorithm used to collapse the n-dimensional grid onto a one dimensional sequence *and* the order in which the axes are considered. The axisNames list belonging to this attribute *MUST* contain only signed axis names which are derived from the axisNames attribute of the associated Grid object (via the “grid” association role.)

1.6.5.3 startSequence

Identifier:	startSequence

Identifies the coordinates of the grid point to be associated with the first record in the values sequence.

1.6.5.4 grid

Identifier:	Extension

Associates this GridValuesMatrix with a geometric description provided by the Grid object. The extent attribute of the associated Grid object must be synchronized with the extent attribute of this object.

1.6.5.5 extent

Identifier:	Extension

Returns the limits of a section of the grid. The envelope contains the low and high coordinates of the minimal envelope that can contain the grid.

1.6.6 GridPositioning

Identifier:	Extension

This is an abstract supertype used to form the Positioning association between Grid and either RectifiedGrid or ReferencableGrid. Implementors should never make an instantiable implementation of this interface.

The two child interfaces represent different levels of complexity for the referencing of gridded data. A RectifiedGrid object is capable of transforming coordinates through a simple affine transformation. A ReferencableGrid object encapsulates an operation of arbitrary complexity.

This type does not exist in ISO 19123. Reference Figure 1 as the defining class diagram.

1.6.6.1 coordinateReferenceSystem

Identifier:	Extension

Specifies the coordinate system into which this object transforms coordinates. ISO 19123 only specifies this association on the ReferenceableGrid type, but it is promoted to this superclass because it is required by both ReferenceableGrid and RectifiedGrid.

1.6.6.2 grid

Identifier:	Extension

Associates this GridPositioning object with a geometric description provided by the Grid object.

1.6.6.3 operation

Identifier:	Extension

Associates this GridPositioning object with descriptive information about the coordinate operation it implements. A RectifiableGrid (or child thereof) will be associated with a coordinate conversion operation, and a ReferencableGrid will be associated with a coordinate transformation operation. All operations include a reference to a MathTransform object, which actually performs the corodinate conversion. The targetCRS association of the operation attribute is considered mandatory in this context.

1.6.6.4 inverseOperation

Identifier:	Extension

Associates this GridPositioning object with descriptive information about the coordinate operation it implements. A RectifiableGrid (or child thereof) will be associated with a coordinate conversion operation, and a ReferencableGrid will be associated with a coordinate transformation operation. All operations include a reference to a MathTransform object, which actually performs the corodinate conversion. The targetCRS association of the inverseOperation attribute is considered mandatory in this context.

This attribute shall represent the Operation which is the inverse of the operation attribute.

1.6.7 RectifiableGrid

Identifier:	Extension

This abstract class represents a general coordinate conversion algorithm to be applied to the grid. In the special case where the coordinate conversion is affine, see RectifiedGrid (section 1.6.8.) This class defines the required convertCoordinates and inverseConvertCoordinates methods required by the RectifiableGrid interface and provides access to the MathTransform object associated with the algorithm. Children of this class need only supply the Conversion object (stored in the inherited “operation” attribute) to produce a functional coordinate conversion object.

1.6.7.1 conversion

Identifier:	Extension

This association shall link the RectifiableGrid class with the coordinate conversion object which defines the coordinate operation to be performed. This conversion object shall be identical to the inherited “operation” attribute.

1.6.7.2 inherited operation attribute

Identifier:	Extension

This attribute shall contain only the “Conversion” subtype of the “Operation” interface, signifying that RectifiableGrid and children represent only *coordinate conversions* as defined by ISO 19111. This attribute shall be identical to the conversion attribute.

1.6.7.3 inverseConversion

Identifier:	Extension

This association shall link the RectifiableGrid class with the coordinate conversion object which defines the inverse coordinate operation to be performed. This conversion object shall be identical to the inherited “inverseOperation” attribute.

1.6.7.4 inherited inverseOperation attribute

Identifier:	Extension

This attribute shall contain only the “Conversion” subtype of the “Operation” interface, signifying that RectifiableGrid and children represent only *coordinate conversions* as defined by ISO 19111.   This attribute shall be identical to the conversion attribute.

1.6.7.5 convertCoordinates

Identifier:	coordConv
Specified Class:	RectifiedGrid

Converts grid coordinates through an affine transform to a direct position. This is an adapter method for the MathTransform object's transform() method. The MathTransform object used in the conversion is associated with the “conversion” and “operation” attributes.

1.6.7.6 inverseConvertCoordinates

Identifier:	invCoordConv
Specified Class:	Rectified Grid

Converts through an affine transform a direct position to the grid coordinates of the nearest grid point. This is an adapter method for the MathTransform object's transform() method. The MathTransform object used in the conversion is associated with the “inverseConversion” and “inverseOperation” attributes.

1.6.7.7 gridToCoordinateSystem

Identifier:	gridToCoordinateSystem
Specification:	OGC 01-004
Obligation:	Optional

This optional attribute is specified on the GC_GridGeometry class in the legacy OGC 01-004 specification. It is retained here because it allows the user access to a conversion object which yields non-integer results. This property is derived from the MathTransform object associated with the operation and conversion attributes, and is merely a convenience method.

1.6.8 RectifiedGrid

Identifier:	CV_RectifiedGrid

Encapsulates a coordinate conversion which must be an affine transformation. This class must be associated (via the inherited grid association role) with a grid for which an affine transformation describes the relationship between the grid coordinates and the coordinates of an external coordinate reference system. A rectified grid is defined by an origin in an external coordinate reference system, and a set of offset vectors that specify the direction and distance between grid lines within that external CRS. As such, it implements a special case of a *coordinate conversion*, as defined in ISO 19111.

This type contains some additional geometric information not in the Grid type which is applicable to the special case of a simple affine relationship to a particular external coordinate system.

The defined constraints applicable to this class:

1. The dimension of the grid shall be less than or equal to the dimension of the coordinate reference system of the point that is the origin.
2. The number of offset vectors shall equal the dimension of the grid.
3. The dimension of all offset vectors shall equal the dimension of the coordinate reference system, even if an offset vector is aligned with an axis of the external coordinate system.

This object, like the ReferenceableGrid object specified in section 1.5.4, does not specify whether the transformation is applied to the grid point or the sample point. However, unlike the ReferenceableGrid, if the relationship between sample point and grid point is a simple constant offset, then this class may be configured for either case because both are affine relations from grid coordinates to world coordinates. Failure to carefully specify the assumptions made about this relationship will lead to misregistration. If the relationship between grid point and sample point is not a simple constant offset, then one set of points has an affine relationship with the external CRS and the other does not. It then becomes *critical* for the implementation to carefully specify which coordinates (grid points or sample points) are transformed to the external CRS.

1.6.8.1 origin

Identifier:	origin

Returns the origin of the rectified grid in an external coordinate reference system. ISO 19123 requires that the *Coordinate Reference System* association of the origin indicate the CRS of all coordinates generated by this object. This implementation specification retains this requirement and further requires that the inherited coordinateReferenceSystem attribute be derived from the *Coordinate Reference System* association of the origin.

1.6.8.2 inherited coodinateReferenceSystem attribute

Identifier:	Extension

In the context of the RectifiedGrid type, the coordinateReferenceSystem attribute inherited from GridPositioning shall be derived from the *Coordinate Reference System* association of the origin.

1.6.8.3 offsetVectors

Identifier:	offsetVectors

Returns the offset vectors that determine the grid spacing in each direction. The vectors are defined in terms of the external coordinate reference system.

1.6.8.4 inherited conversion association role

Identifier:	Extension

The conversion defined by this object is an affine transformation defined by the origin and offset vectors attributes.

1.6.8.5 inherited inverseConversion association role

Identifier:	Extension

The inverseConversion defined by this object is an affine transformation defined by the origin and offset vectors attributes.

1.6.9 SequenceRule

Parameterizes the algorithm which reduces the multidimensional grid to the one-dimensional sequence of Records. This type indicates the algorithm used, the order in which the axes are considered, and the direction along each axis.

1.6.9.1 type

Indicates the algorithm used in the multidimensional to one-dimensional mapping. The default algorithm is "linear".

1.6.9.2 scanDirection

Returns a list of signed axis names that indicates the order in which grid points shall be mapped to position within the sequence of records of feature attribute values. An additional element may be included in the list to allow for interleaving of feature attribute values.

The sign of each axis shall be represented by the first character of the string. If The first character of string is either a "+" or a "-", this explicitly specifies direction along the axis whose name comprises the remainder of the string. If the first character is neither a "+" nor a "-", then "+" shall be assumed.

The axis names in this list must match the axis names of the grid with which the containing GridValuesMatrix is associated. The responsibility for ensuring that this requirement is satisfied rests on the GridValuesMatrix implementation.

1.7 Modeling Pre-work

1.7.1 Classification of GeoAPI methods

TODO: Use the following spreadsheet object to classify the GeoAPI methods. Double-click the spreadsheet (and not the frame) to edit..

GeoAPI Package: org.opengis.xxx

GeoAPI Class/Interface:

Method	Classification (A, R, O)	Type

Table 2: GeoAPI Method Classification for Basic Grid Elements.

1.7.2 Interface and Data Type Proxies

TODO: Use the following spreadsheet object to list the Interface and Data Type proxies required for this implementation effort.

GeoAPI Package:

Interface Proxies	Data Type Proxies

Table 3: Proxy types for Basic Grid Elements

1.8 Test cases

TODO: Describe potential test cases and indicate whether they have been implemented...

Name	***Done?***	***Description***
	y/n	

Table 4: Potential Test Cases for Basic Grid Elements