TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0 PREFACE</td>
<td>9</td>
</tr>
<tr>
<td>0.1 SUBMITTING COMPANIES</td>
<td>9</td>
</tr>
<tr>
<td>0.2 SUBMISSION CONTACT POINTS</td>
<td>9</td>
</tr>
<tr>
<td>0.3 DOCUMENT CONVENTIONS</td>
<td>9</td>
</tr>
<tr>
<td>0.4 REVISION HISTORY</td>
<td>10</td>
</tr>
<tr>
<td>0.5 EDITORIAL NOTES</td>
<td>10</td>
</tr>
<tr>
<td>0.6 CHANGES TO THE ABSTRACT SPECIFICATION</td>
<td>10</td>
</tr>
<tr>
<td>0.7 DEPENDENCIES ON OTHER OGC SPECIFICATIONS</td>
<td>10</td>
</tr>
<tr>
<td>1 SPECIFICATION OVERVIEW</td>
<td>10</td>
</tr>
<tr>
<td>2 ARCHITECTURE</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Grid Coverages</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1 Grid Coverage Characteristics</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2 Grid Coordinate System</td>
<td>12</td>
</tr>
<tr>
<td>2.1.3 Grid Coverage Tiling</td>
<td>12</td>
</tr>
<tr>
<td>2.1.4 Color Palettes</td>
<td>12</td>
</tr>
<tr>
<td>2.1.5 Interoperability</td>
<td>12</td>
</tr>
<tr>
<td>2.2 CV_Coverage</td>
<td>13</td>
</tr>
<tr>
<td>2.3 GC_GridCoverage</td>
<td>14</td>
</tr>
<tr>
<td>2.4 GC_CoverageExchange</td>
<td>15</td>
</tr>
<tr>
<td>2.5 GP_GridAnalysis (optional)</td>
<td>16</td>
</tr>
<tr>
<td>2.6 GP_GridCoverageProcessor (optional)</td>
<td>17</td>
</tr>
<tr>
<td>2.7 Interfaces Supporting Coverage (CV) Package</td>
<td>18</td>
</tr>
<tr>
<td>2.7.1 CV_SampleDimension</td>
<td>18</td>
</tr>
<tr>
<td>2.8 Interfaces/Datatypes Supporting Grid Coverage (GC) Package</td>
<td>18</td>
</tr>
<tr>
<td>2.8.1 GC_GridPacking</td>
<td>18</td>
</tr>
<tr>
<td>2.8.2 GC_GridGeometry</td>
<td>18</td>
</tr>
<tr>
<td>2.8.3 GC_GridRange</td>
<td>18</td>
</tr>
<tr>
<td>2.8.4 GC_ParameterInfo</td>
<td>18</td>
</tr>
<tr>
<td>2.8.5 GC_Parameter</td>
<td>18</td>
</tr>
<tr>
<td>2.8.6 GC_Format</td>
<td>18</td>
</tr>
<tr>
<td>2.9 Interfaces Supporting Grid Coverage Processing (GP) Package</td>
<td>18</td>
</tr>
<tr>
<td>2.9.1 GP_Operation</td>
<td>18</td>
</tr>
<tr>
<td>3 COMPONENT SERVICES</td>
<td>19</td>
</tr>
<tr>
<td>3.1 GRID COMPONENTS CLASSES AND INTERFACES</td>
<td>19</td>
</tr>
<tr>
<td>3.1.1 CV_Coverage</td>
<td>19</td>
</tr>
<tr>
<td>3.1.1.1 Attributes</td>
<td>19</td>
</tr>
<tr>
<td>3.1.1.1.1 +numSampleDimensions : Integer</td>
<td>19</td>
</tr>
</tbody>
</table>
3.1.1.1.2 +dimensionNames : Sequence<CharacterString>  
3.1.1.1.3 +numSource : Integer  
3.1.1.1.4 +metadataNames : Sequence <CharacterString>  
3.1.1.1.5 +coordinateSystem : CS_CoordinateSystem  
3.1.1.1.6 +envelope : PT_Envelope  
3.1.1.2 getSampleDimension  
3.1.1.3 getSource  
3.1.1.4 getMetadataValue  
3.1.1.5 evaluate  
3.1.1.6 evaluateAsBoolean  
3.1.1.7 evaluateAsByte  
3.1.1.8 evaluateAsInteger  
3.1.1.9 evaluateAsDouble  
3.1.2 GC_GridCoverage  
3.1.2.1 Attributes  
3.1.2.1.1 +numSource : Integer  
3.1.2.1.2 +getDataBlockAsXXX  
3.1.2.1.3 +getPackedDataBlock  
3.1.2.1.4 +getOverview  
3.1.2.1.5 +getOverviewGridGeometry  
3.1.2.2 getMetadataValue  
3.1.2.3 getOverview  
3.1.2.4 getOverviewGridGeometry  
3.1.2.5 getG Veranstaltungsprogramm  
3.1.2.6 getOverviewGridGeometry  
3.1.2.7 getG Veranstaltungsprogramm  
3.1.3 GC_GridCoverageExchange  
3.1.3.1 Attributes  
3.1.3.1.1 +numOverview : Integer  
3.1.3.1.2 +metadataNames : Sequence <CharacterString>  
3.1.3.2 +getFormat  
3.1.3.3 +getMetadataValue  
3.1.3.4 createFromName  
3.1.3.5 listSubNames  
3.1.3.6 createFromSubName  
3.1.3.7 exportTo  
3.1.3.8 move  
3.1.4 GP_GridAnalysis  
3.1.4.1 histogram  
3.1.4.2 minValue  
3.1.4.3 maxValue  
3.1.4.4 meanValue  
3.1.4.5 medianValue  
3.1.4.6 modeValue  
3.1.4.7 stdDev  
3.1.4.8 correlation  
3.1.5 GP_GridCoverageProcessor  
3.1.5.1 Attributes  
3.1.5.1.1 +metadataNames : Sequence <CharacterString>  
3.1.5.1.2 +numOperations : Integer  
3.1.5.2 getMetadataValue  
3.1.5.3 getOperation  
3.1.5.4 Analyze  
3.1.5.5 doOperation  
3.1.6 Supporting Interfaces in the Coverage (CV) Package  
3.1.6.1 CV_SampleDimension  
3.1.6.1.1 Attributes
3.1.6.1.1 +description : CharacterString  
3.1.6.1.2 +sampleDimensionType : CV_SampleDimensionType  
3.1.6.1.3 +categoryNames : Sequence<CharacterString>  
3.1.6.1.4 +colorInterpretation : CV_ColorInterpretation  
3.1.6.1.5 +paletteInterpretation : CV_PaletteInterpretation  
3.1.6.1.6 +palette : Sequence<Sequence<Integer>>  
3.1.6.1.7 +metadataNames : Sequence<CharacterString>  
3.1.6.1.8 +metadataValue : Object  
3.1.6.1.9 +palette :  Sequence<CharacterString>  
3.1.6.1.10 +units: CS_Unit  
3.1.6.1.11 +offset : Double  
3.1.6.1.12 +scale : Double  
3.1.6.1.13 +noDataValue : Sequence<Number>  
3.1.6.1.14 +defaultValue : Object  
3.1.6.1.15 +maximumValue : CharacterString  
3.1.6.1.16 +minimumValue : CharacterString  
3.1.7  Supporting Interfaces and Structures in the Grid Coverage (GC) Package  
3.1.7.1 GC_GridPacking  
3.1.7.2.1 Attributes  
3.1.7.2 GC/GridGeometry  
3.1.7.3 GC/GridRange  
3.1.7.4 GC/ParameterInfo  
3.1.7.5 GC/Parameter  
3.1.7.6 GC/Format  
3.1.7.7 GC/ValueInBytePacking  
3.1.7.8 GC/ByteInValuePacking  
3.1.8  Supporting Interfaces in Grid Coverage Processing (GP) Package  
3.1.8.1 GP/Operation
3.1.8.1.1 Attributes
3.1.8.1.1.1 +name : CharacterString
3.1.8.1.1.2 +description : CharacterString
3.1.8.1.1.3 +vendor : CharacterString
3.1.8.1.1.4 +docURL : String
3.1.8.1.1.5 +version : String
3.1.8.1.1.6 +numSources : Integer
3.1.8.1.1.7 +numParameters : Integer

3.1.8.1.2 getParameterInfo

3.2 Well-known Binary Representations
3.2.1 WKBGeoTIFF

3.3 GP_GridCoverageProcessor Operations
3.3.1 Gray Scale Threshold
3.3.2 Image Enhancements
3.3.2.1 Linear Enhancement
3.3.2.2 Root Enhancement
3.3.2.3 Equalization Enhancement
3.3.2.4 Infrequency Enhancement
3.3.3 Interpolate
3.3.4 Band Ratioing
3.3.5 Spatial Filtering
3.3.5.1 Mean Filter
3.3.5.2 Mode Filter
3.3.5.3 Median Filter
3.3.5.4 Gaussian Filter
3.3.5.5 Laplacian Type 1 Filter
3.3.5.6 Laplacian Type 2 Filter
3.3.6 Select Sample Dimensions
3.3.7 Resample
3.3.8 Sequence
3.3.9 Density Slice

4 REFERENCES

APPENDICES
A.1 GLOSSARY OF TERMS
A.2 SUPPORTING WELL-KNOWN VALUES
A.3 EXCEPTIONS, ERRORS AND ERROR CODES
A.4 CONVERSION RULES FOR GRID VALUES TO SPECIFIC DATA TYPES
A.5 COM IDL SPECIFICATION
A.6 CORBA IDL SPECIFICATION
A.7 SPECIFICATION SEQUENCE DIAGRAMS
A.7.1 SIMPLE GRID COVERAGE CREATION EXAMPLE
A.7.2 GRID COVERAGE PROCESSING USING THE DISCOVERY MECHANISM
TABLE OF FIGURES

Figure 1: Specification Packages
Figure 2: CV_Coverage Package
Figure 3: GC_GridCoverage Package
Figure 4: GC_GridCoverageExchange Interface and its relationships
Figure 5: GP_GridAnalysis Interface is a subclass of GC_GridCoverage
Figure 6: GP_GridCoverageProcessor Interface discovery mechanism
Figure 7: CV_Coverage interface for generic coverages
Figure 8: GC_GridCoverage interface providing access to grid data values
Figure 9: GC_GridCoverageExchange Interface
Figure 10: GP_GridAnalysis Interface. GP_GridAnalysis inherits from GC_GridCoverage
Figure 11: GP_GridCoverageProcessor for performing operations in grid coverages
Figure 12: CV_SampleDimension Interface. Descriptive information for a sample dimension
Figure 13: Context Diagram: CV_SampleDimensionType
Figure 14: Context Diagram: CV_PaletteInterpretation
Figure 15: Context Diagram: CV_ColorInterpretation
Figure 16: GC_GridPacking data type
Figure 17: GC_GridGeometry data type
Figure 18: GC_GridRange data type
Figure 19: GC_ParameterInfo data type
Figure 20: GC_Parameter data type
Figure 21: Context Diagram: GC_Format
Figure 22: Context Diagram: GC_ValueInBytePacking
Figure 23: Context Diagram: GC_ByteInValuePacking
Figure 24: Context Diagram: GC_ParameterType
Figure 25: GP_Operation Interface
Figure 26: Simple grid coverage creation
Figure 27: Grid coverage processing using the discovery mechanism
0.0 PREFACE

This implementation specification refers to The Open GIS Abstract Specification Topic 6: The Coverage Type and its Subtypes, the project document 99-106.

0.1 SUBMITTING COMPANIES

The following companies submitted this implementation specification, in response to the OGC Request Number 5 Core Task Force Working Group A Request For Proposals: Access to Open GIS Coverages, OpenGIS Project Document Number 98-007r1.

- Computer Aided Development Corporation Ltd (Cadcorp)
- Environmental Systems Research Institute (ESRI)
- Intergraph Corporation
- Laser-Scan, Ltd.
- Oracle
- PCI Geomatics Inc

0.2 SUBMISSION CONTACT POINTS

All questions regarding this submission should be directed to:

Louis Burry
PCI Geomatics
490 St. Joseph Blvd., Suite 400
Hull, Quebec, Canada, J8Y 3Y7
burry@pcigeomatics.com

Adam Gawne-Cain
Cadcorp Ltd
4 Fitzroy Square
London, UK W1P 5AH
adam@cadcorpdev.co.uk

John Herring
Oracle Corporation
196 vanBuren Street
Herndon, Virginia 20170
jrherrin@us.oracle.com

0.3 DOCUMENT CONVENTIONS

This submission adheres to OGC conventions for UML modeling and CORBA IDL. The COM IDL has followed the Microsoft guidelines for naming conventions since applications will mix OGC/Non-OGC interfaces. The main difference between the OGC and Microsoft conventions is OGC uses a lower case for the first letter for attributes and operations while Microsoft uses upper case.

This document uses the OGC convention of using a two-letter prefix for packages. The following two-letter prefixes used are:
This specification focuses on grid coverages but effort has been made to ensure generic interfaces and components have been specified at the coverage level for future extensions to other coverage types.

The Essential Model for Grid Coverages lists seven varieties of simple pixel modifications. This submission addresses a subset of these. Radiometric Correction Services and Noise Removal Services have a range of complexities and should be addressed by Image Exploitation SIG.

To allow for other georeferencing methods, the GridMatrixValue class was modified. The members of the class which described the georeferencing of the grid, were moved into another class. This change from the abstract specification allows for support of non-orthorectified grids.

The sequential enumeration methods for accessing grid matrix values were limited to row dominant, column dominant and band order. Morton order and pyramid sequences were not addressed. Pyramid sequence was perceived to be an association of lower resolution grid coverages to the grid coverage. This is addressed as overviews in the submission.

The specification uses interfaces from the Open GIS Coordinate Transformation specification.

Components from the CT and CS package provide the georeferencing information for the grid coverage. The georeferencing include the projection and the mapping of grid to projection coordinate for the grid.

This implementation specification was drafted with the following goals:

- Allow for a range of implementations. This will allow for relatively quick and easy implementations of the specification to more comprehensive ones.
- This specification was designed to promote interoperability between implementations done by data vendors and software vendors providing analysis and grid processing implementations. An analysis implementation can utilize a data access implementation.
This specification is written as a general specification with COM and CORBA implementations in mind. This specification could later be extended to include other Distributed Computing Platforms (DCPs) such as JAVA.

Any implementations of this specification should support creation of coverages from a GeoTIFF raster. GeoTIFF is becoming an industry standard and this is meant to allow for standard test datasets to ensure implementations conform to the specification.

The specification must allow for efficient access to very large datasets. Datasets can be up to several gigabytes in size.

Support N dimensions even though most implementations of the specification will be 2 D.

Extensible framework for image processing and analysis.

Be able to handle a wide range of grid data from raw to thematically classified grid coverages.

Support a variety of color models.

2 ARCHITECTURE

The architecture of the grid coverage specification is composed of three packages as shown in Figure 1. The specification includes a package for the general coverage specification (CV), a package specifically for grid coverages (GC) and a package for grid coverage processing (GP). The GP package is optional and not required for an OGC compliant implementation.

2.1 Grid Coverages

2.1.1 Grid Coverage Characteristics

This Grid Coverage Implementation specification can accommodate grid coverages with the following characteristics:

- Variable number of bits per grid value (bits per pixel).
  - 1, 2, 4 unsigned integer
  - 8, 16, 32 signed, unsigned integer
  - 32, 64 Real
- 1 to N number of bands
- 1 to N dimensions
- For grids with multiple bands, band values can be ordered by dimension. For example, a 2 D grid coverage can be ordered row, column, band (pixel interleaved), row, band, column (line interleaved) or band, row, column (band sequential).
- Support for a variable number of “no data values”
- Various color models: Gray Scale, pseudo color (any size), RGB, CMYK and HSL.
• Support of overviews or pyramids (reduced resolution data sets).

2.1.2 Grid Coordinate System

A grid coverage has a grid coordinate system which allows for addressing individual cells in the grid. Individual cells are centered on the grid points. A grid has an ordering of cell values with the first cell in this ordering having a grid coordinate of 0, 0. A two dimensional grid coverage with 512 rows and 512 columns would have grid coordinates with a range from 0 to 511 rows and 0 to 511 columns.

The gridRange attribute in the GC_GridGeometry structure defines the grid coordinate range.

The number of cells in a dimension of the grid can be calculated as grid range maximum – minimum

2.1.3 Grid Coverage Tiling

This implementation specification has no explicit interfaces to retrieve data values from a grid coverage in tiles. It is expected that the implementation will efficiently retrieve data values when the client implementation retrieves data values on tile boundaries. The optimal size to access a dimension can be determined from the GC_GridCoverage optimumDataBlockSizes attribute

2.1.4 Color Palettes

A coverage can have a color palette. A color palette is valid for only coverages which are integer values (usually bytes). Color palettes supported include gray scale, RGB, HSL and CMYK. The coverage dimension information maps the grid sample dimensions to a color palette or to color model components. For example 8-bit pseudo color TIFF image would be represented as a grid with one sample dimension and a color palette with 256 RGB entries. A 24-bit RGB TIFF file would be represented as a grid with three bands and no color palette. The dimension information would indicate which bands to use for the RGB color components. The following illustrates example values for different color components:

<table>
<thead>
<tr>
<th>Color Palette</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gray Scale</td>
<td>Vector contains one entry which a shade of gray. 0 is black and 255 is white.</td>
</tr>
<tr>
<td>RGB</td>
<td>Vector contains three values: Red Green Blue. Each value ranges from 0 to 255 where 0 is no intensity and 255 is maximum intensity.</td>
</tr>
<tr>
<td>RGBA</td>
<td>Vector contains four values RGBA: Red Green Blue Alpha. The alpha value is the level of transparency of the color. Alpha of 0 is transparent and 255 is opaque.</td>
</tr>
<tr>
<td>HSL</td>
<td>Vector contains three values: Hue Saturation Lightness. Hue is a value from 0 to 360, Saturation is a value from 0 to 100 where 0 is no color intensity and 100 is full color intensity. Lightness is a value from 0 to 100 where 0 is no lightness and 100 is full lightness.</td>
</tr>
<tr>
<td>CMYK</td>
<td>Vector contains four values: Cyan Magenta Yellow Black. Each value ranges from 0 to 255 where 0 is no intensity and 255 is maximum intensity.</td>
</tr>
</tbody>
</table>

2.1.5 Interoperability

To help interoperability, many interfaces contained in this architecture have been made immutable. The assumption is that the objects supporting these interfaces should be immutable. This means that the objects should not change after a factory has created them.
In particular, the following interfaces are assumed to be backed by immutable objects:

\[
\text{CV\_SampleDimension, GC\_Format, GC\_GridCoverageExchange, GP\_Operation, GP\_GridAnalysis, GP\_GridCoverageProcessor}
\]

In a GC\_GridCoverage, only the cell values may change. The size, geometry, interpolation, sequencing etc should never change. However, clients should be aware that the Grid Coverage object behind the GC\_GridCoverage interface may be adapted from other GC\_GridCoverage interfaces, so changing the grid point (pixel) values through one GC\_GridCoverage could change the grid point values in others. To allow client applications to anticipate adapted Grid Coverages being changed indirectly, the source Grid Geometry object(s) for adapted Grid Geometry objects can be traced.

2.2 CV\_Coverage

Abstract interface providing access to an OpenGIS coverage.

The essential property of coverage is to be able to generate a value for any point within its domain. How coverage is represented internally is not a concern. For example consider the following different internal representations of coverage:

1. A coverage may be represented by a set of polygons which exhaustively tile a plane (that is each point on the plane falls in precisely one polygon). The value returned by the coverage for a point is the value of an attribute of the polygon that contains the point.
2. A coverage may be represented by a grid of values. The value returned by the coverage for a point is that of the grid value whose location is nearest the point.
3. Coverage may be represented by a mathematical function. The value returned by the coverage for a point is just the return value of the function when supplied the coordinates of the point as arguments.
4. Coverage may be represented by combination of these. For example, coverage may be represented by a combination of mathematical functions valid over a set of polynomials.

Figure 2 illustrates the CV\_Coverage package. A coverage has a corresponding CV\_SampleDimension for each sample dimension in the coverage.
2.3 GC_GridCoverage

This interface would represent the basic implementation which provides access to grid coverage data. A GC_GridCoverage implementation may provide the ability to update grid values. A basic read-only implementation would be fairly easy to implement. The GC package is illustrated in figure 3.
2.4 GC_CoverageExchange

Support for creation of grid coverages from persistent formats as well as exporting a grid coverage to a persistent formats. For example, it allows for creation of grid coverages from the GeoTIFF Well-known Binary format and exporting to the GeoTIFF file format. Basic implementations only require creation of grid coverages from a file format or resource. The interface given in figure 4 has a discovery mechanism to query the available file formats supported.

Figure 3: GC_GridCoverage Package.
2.5 GP_GridAnalysis (optional)

This optional interface performs analysis of the grid data. Such processing functionality includes histogram calculation, grid coverage covariance and other statistical measurements. This interface is illustrated in figure 5.

GP_GridAnalysis operates on a GC_GridCoverage to create a new GC_GridCoverage.

Figure 4: GC_GridCoverageExchange Interface and its relationships

Figure 5: GP_GridAnalysis Interface is a subclass of GC_GridCoverage.
2.6 GP_GridCoverageProcessor (optional)

This optional interface provides operations for different ways of accessing the grid coverage values as well as image processing functionality. The list of available processing operations is implementation dependent. The interface has a discovery mechanism to determine the available processing operations. This is shown in figure 6.

![Interface Diagram](image)

**Figure 6: GP_GridCoverageProcessor Interface discovery mechanism.**

These processing operations will transform values within a single sample dimension, and leave the values in other sample dimensions unaffected. The modified sample dimension may also change its type (e.g. from CV_4BIT to CV_1BIT). The actual underlying grid data remains unchanged.

The interface has been designed to allow the adaptations to be done in a “pipe-lined” manner. The interface operates on GC_GridCoverage to create new a GC_GridCoverage. The interface does not need to make a copy of the source grid data. Instead, it can return a grid coverage object which applies the adaptations on the original grid coverage whenever a block of data is requested. In this way, a pipeline of several grid coverages can be constructed cheaply.

This interface can perform any of the following:

- Change the number of bands being accessed.
- Change the value sequencing in which the grid values are retrieved.
- Allow re-sampling of the grid coverage for a different geometry. Creating a new GC_GridCoverage with different grid geometry allows for reprojecting the grid coverage to another projection and another georeferencing type, resampling to another cell resolution and subsetting the grid coverage.
- Modify the way the grid values are accessed (filtered, classified...).
- Change the interpolation method used when evaluating points which fall between grid cells.
- Filtering
- Image enhancements.
- etc.
2.7 Interfaces Supporting Coverage (CV) Package

2.7.1 CV_SampleDimension

This interface contains information for an individual sample dimension of coverage. This interface is applicable to any coverage type. For grid coverages, the sample dimension refers to an individual band.

2.8 Interfaces/Datatypes Supporting Grid Coverage (GC) Package

2.8.1 GC_GridPacking

This datatype describes the packing of data values within grid coverages. It includes the packing scheme of data values with less than 8 bits per value within a byte, byte packing (Little Endian / Big Endian) for values with more than 8 bits and the packing of the values within the dimensions.

2.8.2 GC_GridGeometry

This datatype describes the geometry and georeferencing information of the grid coverage. The grid range attribute determines the valid grid coordinates and allows for calculation of grid size. See 2.1.2 Grid Coordinate System for more details. A grid coverage may or may not have georeferencing.

2.8.3 GC_GridRange

This datatype specifies the range of valid coordinates for each dimension of the coverage.

2.8.4 GC_ParameterInfo

This datatype provides information for the parameters required for grid coverage processing operations and grid exchange. This information includes such information as the name of the parameter, parameter description, parameter type etc.

2.8.5 GC_Parameter

This datatype for a parameter required for grid coverage processing operations and grid exchange. The parameter contains the parameter keyword and it’s value.

2.8.6 GC_Format

This interface is a discovery mechanism to determine the formats supported by a GC_GridCoverageExchange implementation. A GC_GridCoverageExchange implementation can support a number of file format or resources.

2.9 Interfaces Supporting Grid Coverage Processing (GP) Package

2.9.1 GP_Operation

This interface provides descriptive information for a grid coverage processing operation. The descriptive information includes such information as the name of the operation, operation description, number of source grid coverages required for the operation etc.
3 COMPONENT SERVICES

3.1 GRID COMPONENTS CLASSES AND INTERFACES

3.1.1 CV_Coverage

This interface is common to all coverage types. Other coverage types are subtypes of coverage. This interface is shown in figure 7.

```
<<Interface>>
CV_Coverage
+ numSampleDimensions : Integer
+ dimensionNames : Sequence<CharacterString>
+ numSources : Integer
+ coordinateSystem : CS_CoordinateSystem
+ envelope : PT_Envelope
+ getSource(index : Integer) : CV_Coverage
+ getSampleDimension(index : Integer) : CV_SampleDimension
+ getMetadataValue(name : CharacterString) :
+ evaluate(pt : PT_CoordinatePoint) : Sequence<Number>
+ evaluateAsBoolean(pt : PT_CoordinatePoint) : Sequence<Boolean>
+ evaluateAsByte(pt : PT_CoordinatePoint) : Sequence<byte>
+ evaluateAsInteger(pt : PT_CoordinatePoint) : Sequence<Integer>
+ evaluateAsDouble(pt : PT_CoordinatePoint) : Sequence<Double>

<<Interface>>
GC_GridCoverage
(from GC)
```

Figure 7: CV_Coverage interface for generic coverages.

3.1.1.1 Attributes

3.1.1.1.1 +numSampleDimensions : Integer

**Description:** The number of sample dimensions in the coverage. For grid coverages, a sample dimension is a band.

**Type:** Integer {frozen}

3.1.1.1.2 +dimensionNames : Sequence<CharacterString>

**Description:** The names of each dimension in the coverage. Typically these names are “x”, “y”, “z” and “t”. The number of items in the sequence is the number of dimensions in the coverage. Grid coverages are typically 2D (x, y) while other coverages may be 3D (x, y, z) or 4D (x, y, z, t). The number of dimensions of the coverage is the number of entries in the list of dimension names.

**Type:** Sequence<CharacterString> {frozen}
3.1.1.1.3 +numSource : Integer

**Description:** Number of grid coverages which the grid coverage was derived from. This implementation specification does not include interfaces for creating collections of coverages therefore this value will usually be one indicating an adapted grid coverage, or zero indicating a raw grid coverage.

**Type:** Integer \{frozen\}

3.1.1.1.4 +metadataNames : Sequence <CharacterString>

**Description:** List of metadata keywords for a coverage. If no metadata is available, the sequence will be empty.

**Type:** Sequence<CharacterString> \{frozen\}

3.1.1.1.5 +coordinateSystem : CS_CoordinateSystem

**Description:** This specifies the coordinate system used when accessing a coverage or grid coverage with the “evaluate” methods. It is also the coordinate system of the coordinates used with the math transform (see GC_GridGeometry attribute - gridToCoordinateSystem: CT_MathTransform). This coordinate system is usually different than the grid coordinate system of the grid. A grid coverage can be accessed (re-projected) with new coordinate system with the GP_GridCoverageProcessor component. In this case, a new instance of a grid coverage is created.

Note: If a coverage does not have an associated coordinate system, coordinateSystem will be NULL. The gridToCoordinateSystem attribute for GC_GridGeometry should also be NULL if coordinateSystem is NULL.

**Type:** CS_CoordinateSystem \{frozen\}

3.1.1.1.6 +envelope : PT_Envelope

**Description:** The bounding box for the coverage domain in coordinate system coordinates

For grid coverages, the grid cells are centered on each grid coordinate. The envelope for a 2-D grid coverage includes the following corner positions.

- (Minimum row - 0.5, Minimum column - 0.5) for the minimum coordinates
- (Maximum row - 0.5, Maximum column - 0.5) for the maximum coordinates

Note: If a grid coverage does not have any associated coordinate system, the minimum and maximum coordinate points for the envelope will be empty sequences.

**Type:** PT_Envelope \{frozen\}

3.1.1.2 getSampleDimension

**Prototype:** getSampleDimension(index : Integer) : CV_SampleDimension

**Class:** CV_Coverage

**Description:** Retrieve Sample dimension information for the coverage. For a grid coverage, a sample dimension is a band. The sample dimension information include such things as description, data type of the value
(bit, byte, integer…), the no data values, minimum and maximum values and a color table if one is associated with the dimension. A coverage must have at least one sample dimension.

**Inputs:**

index   Index for sample dimension to retrieve. Indices are numbered 0 to (n-1).

**OutPuts:**  None

**Returns:**  CV_SampleDimension

**Exceptions:**  CV_InvalidIndex

### 3.1.1.3 getSource

**Prototype:**  getSource(sourceDataIndex : Integer) : GC_GridCoverage

**Class:**  CV_Coverage

**Description:**  Returns the source data for a grid coverage. If the GC_GridCoverage was produced from an underlying dataset (by createFromName() or createFromSubName for instance) the NumSource property should be zero, and this method should not be called. If the GC_GridCoverage was produced using GP_GridCoverageProcessor then it should return the source grid coverage of the one used as input to GP_GridCoverageProcessor. In general the source() method is intended to return the original GC_GridCoverage on which it depends. This is intended to allow applications to establish what GC_GridCoverages will be affected when others are updated, as well as to trace back to the "raw data".

**Inputs:**

sourceDataIndex  Source grid coverage index. Indexes start at 0.

**OutPuts:**  None

**Returns:**  GC_GridCoverage

**Exceptions:**  CV_InvalidIndex

### 3.1.1.4 getMetadataValue

**Prototype:**  getMetadataValue(name : CharacterString) : CharacterString

**Class:**  CV_Coverage

**Description:**  Retrieve the metadata value for a given metadata name.

**Inputs:**

name  Metadata keyword for which to retrieve metadata.

**OutPuts:**  None

**Returns:**  CharacterString

**Exceptions:**  CV_MetadataNameNotFound

### 3.1.1.5 evaluate

**Prototype:**  evaluate(point : PT_CoordinatePoint) : Vector

**Class:**  CV_Coverage

**Description:**  Return the value vector for a given point in the coverage. A value for each sample dimension is included in the vector. The default interpolation type used when accessing grid values for points which fall between grid cells is nearest neighbor. The coordinate system of the point is the same as the grid coverage coordinate system (specified by the coordinateSystem attribute).
Inputs:

point  Point at which to find the grid values.

OutPuts: None
Returns: Vector
Exceptions: CV_PointOutsideCoverage

3.1.1.6 evaluateAsBoolean

Prototype: evaluateAsBoolean(point : PT_CoordinatePoint) : Sequence<Boolean>
Class: CV_Coverage
Description: Return a sequence of Boolean values for a given point in the coverage. A value for each sample dimension is included in the sequence. The default interpolation type used when accessing grid values for points which fall between grid cells is nearest neighbor. The coordinate system of the point is the same as the grid coverage coordinate system.

For converting a value to Boolean, see APPENDICE 5 CONVERSION RULES FOR VALUES TO LANGUAGE SPECIFIC DATA TYPES.

Parameters

point  Point at which to find the coverage values.

OutPuts: None
Returns: Sequence<Boolean>
Exceptions: CV_PointOutsideCoverage

3.1.1.7 evaluateAsByte

Prototype: evaluateAsByte(point : PT_CoordinatePoint) : Sequence<Byte>
Class: CV_Coverage
Description: Return a sequence of unsigned byte values for a given point in the coverage. A value for each sample dimension is included in the sequence. The default interpolation type used when accessing grid values for points which fall between grid cells is nearest neighbor. The coordinate system of the point is the same as the grid coverage coordinate system.

For converting a value to Byte, see APPENDICE 5 CONVERSION RULES FOR VALUES TO LANGUAGE SPECIFIC DATA TYPES.

Inputs:

point  Point at which to find the coverage values.

OutPuts: None
Returns: Sequence<Byte>
Exceptions: CV_PointOutsideCoverage

3.1.1.8 evaluateAsInteger

Prototype: evaluateAsInteger(point : PT_CoordinatePoint) : Sequence<Integer>
**Class:** CV_Coverage

**Description:** Return a sequence of integer values for a given point in the coverage. A value for each sample dimension is included in the sequence. The default interpolation type used when accessing grid values for points which fall between grid cells is nearest neighbor. The coordinate system of the point is the same as the grid coverage coordinate system.

For converting a value to Integer, see APPENDICE 5 CONVERSION RULES FOR VALUES TO LANGUAGE SPECIFIC DATA TYPES.

For integral values which have less than 32 bits, values are padded to 32 bits.

**Inputs:**
point Point at which to find the grid values.

**OutPuts:** None

**Returns:** Sequence<Integer>

**Exceptions:** CV_PointOutsideCoverage

### 3.1.1.9 evaluateAsDouble

**Prototype:** evaluateAsDouble(point : PT_CoordinatePoint) : Sequence<Double>

**Class:** CV_Coverage

**Description:** Return an sequence of double values for a given point in the coverage. A value for each sample dimension is included in the sequence. The default interpolation type used when accessing grid values for points which fall between grid cells is nearest neighbor. The coordinate system of the point is the same as the grid coverage coordinate system.

For converting a value (v) to double (d), see APPENDICE 5 CONVERSION RULES FOR GRID VALUES TO LANGUAGE SPECIFIC DATA TYPES.

**Inputs:**
point Point at which to find the grid values.

**OutPuts:** None

**Returns:** Sequence<Double>

**Exceptions:** CV_PointOutsideCoverage

### 3.1.2 GC_GridCoverage

The GC_GridCoverage interface provides basic access to grid data values. This interface is shown in figure 8. Each band in an image is represented as a sample dimension.
Figure 8: GC_GridCoverage interface providing access to grid data values.

3.1.2.1 Attributes

3.1.2.1.1 +dataEditable : Boolean

**Description:** True if grid data can be edited.

**Type:** Boolean {frozen}
3.1.2.1.2 +gridPacking : GC_GridPacking

**Description:** Information for the packing of grid coverage values.

**Type:** GC_GridPacking {frozen}

3.1.2.1.3 +gridGeometry : GC_GridGeometry

**Description:** Information for the grid coverage geometry. Grid geometry includes the valid range of grid coordinates and the georeferencing.

**Type:** GC_GridGeometry {frozen}

3.1.2.1.4 +numOverviews : Integer

**Description:** Number of predetermined overviews for the grid.

**Type:** Integer {frozen}

3.1.2.1.5 +optimalDataBlockSizes : Sequence<Integer>

**Description:** Optimal size to use for each dimension when accessing grid values. These values together give the optimal block size to use when retrieving grid coverage values. For example, a client application can achieve better performance for a 2-D grid coverage by reading blocks of 128 by 128 if the grid is “tiled” into blocks of this size. The sequence is ordered by dimension.

**Note:** If the implementation does not have optimal sizes the sequence will be empty.

**Type:** Sequence<Integer> {frozen}

3.1.2.2 getOverviewGridGeometry

**Prototype:** getOverviewGridGeometry(overviewIndex : Integer) : GC_GridGeometry

**Class:** GC_GridCoverage

**Description:** Return the grid geometry for an overview.

**Inputs:**

overviewIndex Overview index for which to retrieve grid geometry. Indices start at 0.

**OutPuts:** None

**Returns:** GC_GridGeometry

**Exceptions:** GC_InvalidIndex

3.1.2.3 getOverview

**Prototype:** getOverview(overviewIndex : Integer) : GC_GridCoverage

**Class:** GC_GridCoverage

**Description:** Returns a pre-calculated overview for a grid coverage. The overview indices are numbered from 0 to numberOverviews – 1.
The overviews are ordered from highest (index 0) to lowest (numberOverviews \(-\) 1) resolution. Overview grid coverages will have overviews which are the overviews for the grid coverage with lower resolution than the overview.

For example:

A 1 meter grid coverage with 3, 9, and 27 meter overviews will be ordered as follows:

<table>
<thead>
<tr>
<th>Index</th>
<th>resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
</tr>
</tbody>
</table>

The 3 meter overview will have 2 overviews as follows:

<table>
<thead>
<tr>
<th>Index</th>
<th>resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>27</td>
</tr>
</tbody>
</table>

overviewIndex | Index of grid coverage overview to retrieve. Indexes start at 0.

**OutPuts:** None

**Returns:** GC_GridCoverage

**Exceptions** GC_InvalidIndex

### 3.1.2.4 getDataBlockAsXXX

**Prototype:**

- getDataBlockAsBoolean(gridRange : GC_GridRange) : Sequence<Boolean>
- getDataBlockAsByte(gridRange : GC_GridRange) : Sequence<Byte>
- getValueBlockAsInteger(gridRange : GC_GridRange) : Sequence<Integer>
- getValueBlockAsDouble(gridRange : GC_GridRange) : Sequence<Double>

**Class:** GC_GridCoverage

**Description:** Return a sequence of strongly typed values for a block. A value for each sample dimension will be returned.

For CORBA, the sequencing order of the values in the sequence will follow the rules defined in bandPacking GC_GridPacking.

For COM, the return value is an N + 1 dimensional safe-array, with dimensions (sample dimension, dimension \(n\), dimension \(n-1\), … dimension 1). For 2 dimensional grid coverages, this safe array will be accessed as (sample dimension, column, row). The index values will be based from 0. The indices in the returned N Dimensional safe array will need to be offset by gridRange minimum coordinates to get equivalent grid coordinates.

The requested grid range must satisfy the following rules for each dimension of the grid coverage:

\[
\text{Min grid coordinate} \leq \text{grid range minimum} \leq \text{grid range maximum} \leq \text{maximum grid coordinate}
\]

The number of values returned will equal:

\[
(\text{Max}_1 - \text{Min}_1 + 1) * (\text{Max}_2 - \text{Min}_2 + 1) * \ldots * (\text{Max}_n - \text{Min}_n + 1) * \text{numberSampleDimensions}
\]

Where

- \(\text{Min}\) is the minimum ordinate in the grid range
- \(\text{Max}\) is the maximum ordinate in the grid range
N is the number of dimensions in the grid coverage

**Inputs:**
- gridRange: Grid range for block of data to be accessed.

**OutPuts:**
- None

**Returns:**
- Sequence<XXX>

**Exceptions:**
- GC_InvalidRange

### 3.1.2.5 `getPackedDataBlock`

**Prototype:**
```
getcPackedDataBlock(gridRange : GC_GridRange) : Sequence<Byte>
```

**Class:** GC_GridCoverage

**Description:** Return a block of grid coverage data for all sample dimensions. A value for each sample dimension will be returned. This operation provides efficient access of the grid values. The sequencing order of the values in the sequence will follow the rules given by valueInBytePacking and bandPacking defined in GC_GridPacking.

The requested grid range must satisfy the following rules for each dimension of the grid coverage:

\[
\text{Min grid coordinate} \leq \text{grid range minimum} \leq \text{grid range maximum} \leq \text{maximum grid coordinate}
\]

The sequence of bytes returned will match the data type of the dimension. For example, a grid with one 16 bit unsigned (CV_16BIT_U) sample dimension will return 2 bytes for every cell in the block.

**Byte padding Rules for grid values of less than 8 bits**

For 2D grid coverages, padding is to the nearest byte for the following cases:

- **For PixelInterleaved**
  - For grids with multiple sample dimensions, padding occurs between pixels for each change in dimension type.

- **For LineInterleaved**
  - Padding occurs at the end of each row or column (depending on the valueSequence of the grid).

- **For BandSequential**
  - Padding occurs at the end of every sample dimension.

For grid values smaller than 8 bits, their order within each byte is given by the valueInBytePacking defined in GC_GridPacking.

For grid values bigger than 8 bits, the order of their bytes is given by the byteInValuePacking defined in GC_GridPacking.

**Inputs:**
- gridRange: Grid range for block of data to be accessed.

**OutPuts:**
- None

**Returns:**
- Sequence<Byte>

**Exceptions:**
- GC_InvalidRange
3.1.2.6 setDataBlockAsXXX

**Prototype:**

- `setDataBlockAsBoolean(gridRange : GC_GridRange, values : Sequence<Boolean>): Null`
- `setDataBlockAsByte(gridRange : GC_GridRange, values : Sequence<Byte>): Null`
- `setDataBlockAsInteger(gridRange : GC_GridRange, values : Sequence<Integer>): Null`
- `setDataBlockAsDouble(gridRange : GC_GridRange, values : Sequence<Double>): Null`

**Class:** GC_GridCoverage

**Description:** Set a block of strongly typed values for all sample dimensions.

The requested grid range must satisfy the following rules for each dimension of the grid coverage:

\[
\text{Min grid coordinate} \leq \text{grid range minimum} \leq \text{grid range maximum} \leq \text{maximum grid coordinate}
\]

The number of values must equal:

\[
(\text{Max}_1 - \text{Min}_1 + 1) \times (\text{Max}_2 - \text{Min}_2 + 1) \times \ldots \times (\text{Max}_N - \text{Min}_N + 1) \times \text{numberSampleDimensions}
\]

Where
- \( \text{Min} \) is the minimum ordinate in the grid range
- \( \text{Max} \) is the maximum ordinate in the grid range
- \( N \) is the number of dimensions in the grid coverage

**Inputs:**

- `gridRange`: Grid range for block of data to be accessed.
- `values`: Sequence of grid values for the given region.

**OutPuts:** None

**Returns:** Null

**Exceptions:** GC_InvalidRange, GC_GridNotEditable

3.1.2.7 setPackedDataBlock

**Prototype:**

`setPackedDataBlock(gridRange : GC_GridRange, values : Sequence<Byte>): Null`

**Class:** GC_GridCoverage

**Description:** Set a block of grid coverage data for all sample dimensions. See `GetDataBlock` for details on how to pack the values.

The requested grid range must satisfy the following rules for each dimension of the grid coverage:

\[
\text{Min grid coordinate} \leq \text{grid range minimum} \leq \text{grid range maximum} \leq \text{maximum grid coordinate}
\]

For byte padding rules see `getDataBlock`.

**Inputs:**

- `gridRange`: Grid range for block of data to be accessed.
values    Sequence of grid values for the given region.

OutPuts: None
Returns: None
Exceptions: GC_InvalidRange
          GC_GridNotEditable

3.1.3 GC_GridCoverageExchange

The GC_GridCoverageExchange creates instances of grid coverages and can export grid coverage to persistent file formats. This interface is illustrated in figure 9.

3.1.3.1 Attributes

3.1.3.1.1 +numFormats : Integer

Description: The number of formats supported by the GC_GridCoverageExchange.
Type: Integer {frozen}

3.1.3.1.2 +metadataNames : Sequence<CharacterString>

Description: List of metadata keywords for the interface. If no metadata is available, the sequence will be empty.
Type: Sequence<CharacterString> {frozen}

3.1.3.2 getFormat

Prototype: getFormat(index : integer) : GC_Format
Class: GC_GridCoverageExchange
Description: Retrieve information on file formats or resources available with the GC_GridCoverageExchange implementation. Indices start at zero.
Inputs:
index Index for which to retrieve the format information.
3.1.3.3 getMetadataValue

Prototype: getMetadataValue(name : CharacterString) : CharacterString
Class: GC_GridCoverageExchange
Description: Retrieve the metadata value for a given metadata name.

Inputs:
name Metadata keyword for which to retrieve metadata.

OutPuts: None
Returns: CharacterString
Exceptions: GC_MetadataNameNotFound

3.1.3.4 createFromName

Prototype: createFromName(name : CharacterString) : GC_GridCoverage
Class: GC_GridCoverageExchange
Description: Create a new GC_GridCoverage from a grid coverage file. This method is meant to allow implementations to create a GC_GridCoverage from any file format. A implementation can support any number of formats which is determined from the GC_Format interface.

Inputs:
name File name (including path) from which to create a grid coverage interface. This file name can be any valid file name within the underlying operating system of the server or a valid string, such as a URL which specifics a grid coverage. Each implementation must determine if file name is valid for it's own use.

OutPuts: None
Returns: GC_GridCoverage
Exceptions: GC_CANNOTCreateGridCoverage

3.1.3.5 listSubNames

Prototype: listSubNames(name : CharacterString) : Sequence<CharacterString>
Class: GC_GridCoverageExchange
Description: Retrieve the list of grid coverages contained within the given file or resource. Each grid can have a different coordinate system, number of dimensions and grid geometry.

For example, a HDF-EOS file (GRID.HDF) contains 6 grid coverages each having a different projection.

GRID.HDF:
| UTM |
| Geo |
| Polar |
| IGoode |
| SOM |
| Lamaz |
Note: An empty sequence will be returned if no sub names exist.

**Inputs:**

name  File name (including path) from which to retrieve the grid coverage names. This file name can be any valid file name within the underlying operating system of the server or a valid string, such as a URL which specifies a grid coverage. Each implementation must determine if file name is valid for its own use. Implementations can support many different of file formats

**OutPuts:**

None

**Returns:**

Sequence<CharacterString>

**Exceptions:**

None

3.1.3.6 createFromSubName

**Prototype:**

createFromSubName(name : CharacterString, subName : CharacterString) : GC_GridCoverage

**Class:**

GC_GridCoverageExchange

**Description:**

Create a new GC_GridCoverage from a file where the file contains many grid coverages. This method is meant to allow implementations to create a GC_GridCoverage from any file format which contains many grid coverages. An example of such a format is HDF-EOS format.

**Inputs:**

name  File name (including path) from which to create a grid coverage interface. This file name can be any valid file name within the underlying operating system of the server or a valid string, such as a URL which specifics a grid coverage. Each implementation must determine if name is valid for its own use.

subName  Name of grid coverage contained in file name or resource.

**OutPuts:**

None

**Returns:**

GC_GridCoverage

**Exceptions:**

GC_CannotCreateGridCoverage

3.1.3.7 exportTo

**Prototype:**

exportTo(gridCoverage : GC_GridCoverage, fileFormat : CharacterString, fileName : CharacterString, creationOptions : Sequence<GC_Parameter>) : Null

**Class:**

GC_GridCoverageExchange

**Description:**

Export a grid coverage to a persistent file format.

**Inputs:**

gridCoverage  Source grid coverage.

fileFormat  String which indicates exported file format. The file format types are implementation specific. The file format name is determined from the GC_Format interface.

Sample file formats include:

- “GeoTIFF” - GeoTIFF
- “PIX” - PCI Geomatics PIX
- “HDF-EOS” - NASA HDF-EOS
- “NITF” - National Image Transfer Format
- “STDS-DEM” - Standard Transfer Data Standard
Other file format names are implementation dependent.

**fileName**
File name to store grid coverage. This file name can be any valid file name within the underlying operating system of the server.

**createOptions**
Options to use for creating the file. These options are implementation specific are the valid options is determined from the GC_Format interface.

| **OutPuts:** | None |
| **Returns:** | Null |
| **Exceptions:** | GC_FileFormatNotCompatiableWithGridCoverage GC_ErrorExportingGridCoverage GC_InvalidParameterName GC_InvalidParameterValue |

### 3.1.3.8 move

**Prototype:**
```plaintext
```

**Class:** GC_GridCoverageExchange

**Description:**
Create a new coverage with a different coordinate reference system.

**Inputs:**
- **gridCoverage** Source grid coverage.
- **coordsys** Coordinate system of the new grid coverage.
- **gridToCoordinateSystem** Math transform to assign to grid coverage.

| **OutPuts:** | None |
| **Returns:** | GC_GridCoverage |
| **Exceptions:** | None |

### 3.1.4 GP_GridAnalysis

The GP_GridAnalysis interface performs various analysis operations on a grid coverage. This interface is shown in figure 10.

![GP_GridAnalysis Interface](image)

**Figure 10: GP_GridAnalysis Interface.** GP_GridAnalysis inherits from GC_GridCoverage.
3.1.4.1 histogram

Prototype:  
```
histogram(sampleDimension : Integer, minimumEntryValue : Number,
maximumEntryValue : Number, numberEntries : Integer) :
Sequence<Integer>
```

Class:  
GP_GridAnalysis

Description:  
Determine the histogram of the grid values for a sample dimension.

Inputs:  
- sampleDimension  Index of sample dimension to be histogrammed.
- minimumEntryValue  Minimum value stored in the first histogram entry.
- maximumEntryValue  Maximum value stored in the last histogram entry.
- numberEntries  Number of entries in the histogram.

Outputs:  
None

Returns:  
Sequence<Integer>

Exceptions:  
GP_InvalidSampleDimension

3.1.4.2 minValue

Prototype:  
```
minValue(sampleDimension : Integer) : Number
```

Class:  
GP_GridAnalysis

Description:  
Determine the minimum grid value for a sample dimension.

Inputs:  
- sampleDimension  Index of sample dimension.

Outputs:  
None

Returns:  
Number

Exceptions:  
GP_InvalidSampleDimension

3.1.4.3 maxValue

Prototype:  
```
maxValue(sampleDimension : Integer) : Number
```

Class:  
GP_GridAnalysis

Description:  
Determine the maximum grid value for a sample dimension.

Inputs:  
- sampleDimension  Index of sample dimension.

Outputs:  
None

Returns:  
Number

Exceptions:  
GP_InvalidSampleDimension
3.1.4.4 meanValue

Prototype: meanValue(sampleDimension : Integer) : Double
Class: GP_GridAnalysis
Description: Determine the mean grid value for a sample dimension.

Inputs:
sampleDimension  Index of sample dimension.

OutPuts: None
Returns: Double
Exceptions: GP_InvalidSampleDimension

3.1.4.5 medianValue

Prototype: medianValue(sampleDimension : Integer) : Number
Class: GP_GridAnalysis
Description: Determine the median grid value for a sample dimension.

Inputs:
sampleDimension  Index of sample dimension.

OutPuts: None
Returns: Number
Exceptions: GP_InvalidSampleDimension

3.1.4.6 modeValue

Prototype: modeValue(sampleDimension : Integer) : Number
Class: GP_GridAnalysis
Description: Determine the mode grid value for a sample dimension.

Inputs:
sampleDimension  Index of sample dimension.

OutPuts: None
Returns: Number
Exceptions: GP_InvalidSampleDimension

3.1.4.7 stdDev

Prototype: stdDev(sampleDimension : Integer) : Double
Class: GP_GridAnalysis
Description: Determine the standard deviation from the mean of the grid values for a sample dimension.

Inputs:
sampleDimension  Index of sample dimension.

OutPuts: None
Returns: Double
Exceptions: GP_InvalidSampleDimension
3.1.4.8 correlation

**Prototype:** correlation() : PT_Matrix  
**Class:** GP_GridAnalysis  
**Description:** Determine the correlation between sample dimensions in the grid.  
**Inputs:** None  
**Output:** None  
**Returns:** PT_Matrix  
**Exceptions:** GP_GridCoverageHasOnlyOneSampleDimension

3.1.5  GP_GridCoverageProcessor

The GP_GridCoverageProcessor allows for different ways of accessing the grid coverage values. This interface is shown in figure 11.

![Diagram of GP_GridCoverageProcessor](image)

**Figure 11:** GP_GridCoverageProcessor for performing operations in grid coverages.

Using one of these operations to change the way the grid is being accessed will not affect the state of the grid coverage controlled by another operation. For example, changing the interpolation method should not affect the number of sample dimensions currently being accessed or value sequence.

3.1.5.1 Attributes

3.1.5.1.1 +metadataNames : Sequence<CharacterString>

**Description:** Retrieve the list of metadata keywords for the interface. An empty list will returned if no metadata is available.  
**Type:** Sequence<CharacterString> {frozen}
3.1.5.1.2 **+numOperations : Integer**

**Description:** The number of operations supported by the GP_GridCoverageProcessor.

**Type:** Integer {frozen}

3.1.5.2 **getMetadataValue**

**Prototype:**
getMetadataValue(name : CharacterString) : CharacterString

**Class:** GP_GridCoverageProcessor

**Description:** Retrieve the metadata value for a given metadata name.

**Inputs:**
name Metadata keyword for which to retrieve metadata.

**OutPuts:** None

**Returns:** CharacterString

**Exceptions:** GP_MetadataNameNotFound

3.1.5.3 **getOperation**

**Prototype:**
getOperation(index : Integer) : GP_Operation

**Class:** GP_GridCoverageProcessor

**Description:** Retrieve a grid processing operation information. The operation information will contain the name of the operation as well as a list of its parameters.

**Inputs:**
Index Index for which to retrieve the operation information.

**OutPuts:** None

**Returns:** GP_Operation

**Exceptions:** GP_InvalidIndex

3.1.5.4 **Analyze**

**Prototype:**
Analyze(gridCoverage: GC_GridCoverage) : GP_GridAnalysis

**Class:** GP_GridCoverageProcessor

**Description:** Creates a GP_GridAnalysis interface from a grid coverage. This allows grid analysis functions to be performed on a grid coverage.

**Inputs:**
gridCoverage Grid coverage on which the analysis will be performed.

**OutPuts:** None

**Returns:** GP_GridAnalysis

**Exceptions:** None

3.1.5.5 **doOperation**

**Prototype:**
doOperation(operationName : String, parameters : Sequence<GC_Parameter>) : GC_GridCoverage
**Class:** GP_GridCoverageProcessor

**Description:** Apply a process operation to a grid coverage.

**Inputs:**
- **operationName** Name of the operation to be applied to the grid coverage.
- **parameters** List of name value pairs for the parameters required for the operation.

**Outputs:** None

**Returns:** GC_GridCoverage

**Exceptions:**
- GP_OperationNotFound
- GP_InvalidParameterName
- GP_InvalidParameterValue

### 3.1.6 Supporting Interfaces in the Coverage (CV) Package

#### 3.1.6.1 CV_SampleDimension

The CV_SampleDimension interface describes the data values for a coverage. For a grid coverage a sample dimension is a band. This interface is shown in figure 12.

![Figure 12: CV_SampleDimension Interface. Descriptive information for a sample dimension.](image)

**3.1.6.1.1 Attributes**

**3.1.6.1.1.1 +description : CharacterString**

**Description:** Sample dimension title or description. This string may be empty if no description is present.

**Type:** CharacterString {frozen}

**3.1.6.1.1.2 +sampleDimensionType : CV_SampleDimensionType**
**Description:** A code value indicating grid value data type. This will also indicate the number of bits for the data type.

**Type:** CV_SampleDimensionType {frozen}

3.1.6.1.1.3 +categoryNames : Sequence<

**Description:** Sequence of category names for the values contained in a sample dimension. This allows for names to be assigned to numerical values. The first entry in the sequence relates to a cell value of zero. For grid coverages, category names are only valid for a classified grid data.

For example:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;Background&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;Water&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Forest&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;Urban&quot;</td>
</tr>
</tbody>
</table>

Note: If no category names exist, an empty sequence is returned.

**Type:** Sequence<

3.1.6.1.1.4 +colorInterpretation : CV_ColorInterpretation

**Description:** Color interpretation of the sample dimension. A sample dimension can be an index into a color palette or be a color model component. If the sample dimension is not assigned a color interpretation the value is CV_Undefined.

**Type:** CV_ColorInterpretation {frozen}

3.1.6.1.1.5 +paletteInterpretation : CV_PaletteInterpretation

**Description:** Indicates the type of color palette entry for sample dimensions which have a palette. If a sample dimension has a palette, the color interpretation must be CV_GrayIndex or CV_PaletteIndex. A palette entry type can be Gray, RGB, CMYK or HLS.

**Type:** CV_PaletteInterpretation {frozen}

3.1.6.1.1.6 +palette : Sequence<Sequence<Integer>>

**Description:** Color palette associated with the sample dimension. A color palette can have any number of colors. See palette interpretation for meaning of the palette entries.

Note: If the grid coverage has no color palette, an empty sequence will be returned.

**Type:** Sequence<Sequence<Integer>> {frozen}

3.1.6.1.1.7 +noDataValue : Sequence<

**Description:** Values to indicate "no data" values for the sample dimension. For low precision sample dimensions, this will often be no "no data values".

Note: If there are no "no data values", the sequence will be empty.

**Type:** Sequence<

3.1.6.1.1.8 +minimumValue : Number
Description: The minimum value occurring in the sample dimension. If this value is not available, this value can be determined from the GP_GridAnalysis minValue operation.

Type: Number {frozen}

3.1.6.1.9 +maximumValue : Number

Description: The maximum value occurring in the sample dimension. If this value is not available, this value can be determined from the GP_GridAnalysis maxValue operation.

Type: Number {frozen}

3.1.6.1.10 +units: CS_Unit

Description: The unit information for this sample dimension. This interface typically is provided with grid coverages which represent digital elevation data.

Type: CS_Unit {frozen}

3.1.6.1.11 +offset : Double

Description: Offset is the value to add to grid values for this sample dimension. This attribute is typically used when the sample dimension represents elevation data. The default for this value is 0.

Type: Double {frozen}

3.1.6.1.12 +scale : Double

Description: Scale is the value which is multiplied to grid values for this sample dimension. This attribute is typically used when the sample dimension represents elevation data. The default for this value is 1.

Type: Double {frozen}

3.1.6.1.13 +metadataNames : Sequence<CharacterString>

Description: The list of metadata keywords for a sample dimension. If no metadata is available. The sequence will be empty.

Type: Sequence<CharacterString> {frozen}

3.1.6.1.2 getMetadataValue

Prototype: getMetadataValue(name : CharacterString) : CharacterString

Class: CV_SampleDimension

Description: Retrieve the metadata value for a given metadata name.

Inputs:

name Metadata keyword for which to retrieve metadata.
3.1.6.2 CV_SampleDimensionType

CV_SampleDimensionType specifies the various dimension types for coverage values. For grid coverages, these correspond to band types. This codelist is shown in figure 13.

```
<<CodeList>>
CV_SampleDimensionType
+ CV_1BIT : Integer = 0
+ CV_2BIT : Integer = 1
+ CV_4BIT : Integer = 2
+ CV_8BIT_U : Integer = 3
+ CV_8BIT_S : Integer = 4
+ CV_16BIT_U : Integer = 5
+ CV_16BIT_S : Integer = 6
+ CV_32BIT_U : Integer = 7
+ CV_32BIT_S : Integer = 8
+ CV_32BIT_REAL : Integer = 9
+ CV_64BIT_REAL : Integer = 10
```

Figure 13: Context Diagram: CV_SampleDimensionType

3.1.6.3 CV_PaletteInterpretation

CV_PaletteInterpretation describes the color entry in a color table. See Appendix 3 for specifics on how these type relate to the color entry Well-known Structure. This codelist is shown in figure 14

```
<<CodeList>>
CV_PaletteInterpretation
+ CV_Gray : Integer = 0
+ CV_RGB : Integer = 1
+ CV_CMYK : Integer = 2
+ CV_HLS : Integer = 3
```

Figure 14: Context Diagram: CV_PaletteInterpretation

3.1.6.4 CV_ColorInterpretation

CV_ColorInterpretation specifies the mapping of a band to a color model component. This codelist is shown in figure 15.
The values of CV_ColorInterpretation mean:

- CV_Undefined: Band is not associated with a color model component.
- CV_GrayIndex: Band is an index into a lookup table.
- CV_PaletteIndex: Band is a color index into a color table.
- CV_RedBand: Bands correspond to RGB color model components. AlphaBand may or may not be present.
- CV_GreenBand
- CV_BlueBand
- CV_AlphaBand
- CV_HueBand: Bands correspond to HSL color model.
- CV_SaturationBand
- CV_LightnessBand
- CV_CyanBand: Bands correspond to CMYK color model.
- CV.MagentaBand
- CV.YellowBand
- CV.BlackBand

3.1.7 Supporting Interfaces and Structures in the Grid Coverage (GC) Package

3.1.7.1 GC_GridPacking

The GC_GridPacking data type indicates the packing organization of grid coverage data values. This data type is shown in figure 16.
3.1.7.1.1 Attributes

3.1.7.1.1.1 +byteInValuePacking : GC_ByteInValuePacking

**Description:** Order of bytes packed in values for sample dimensions with greater than 8 bits.

**Type:** GC_ByteInValuePacking {frozen}

3.1.7.1.1.2 +valueInBytePacking : GC_ValueInBytePacking

**Description:** Order of values packed in a byte for CV_1BIT, CV_2BIT and CV_4BIT data types.

**Type:** GC_ValueInBytePacking {frozen}

3.1.7.1.1.3 +bandPacking : Integer

**Description:** Gives the ordinate index for the band. This index indicates how to form a band-specific coordinate from a grid coordinate and a sample dimension number.

This indicates the order in which the grid values are stored in streamed data. This packing order is used when grid values are retrieved using the PackedDataBlock or set using SetPackedDataBlock operations on GC_GridCoverage.

- bandPacking of
  - 0 the full band-specific coordinate is (b, n1, n2...)
  - 1 the full band-specific coordinate is (n1, b, n2...)
  - 2 the full band-specific coordinate is (n1, n2, b...)

Where
- b is band
- n1 is dimension 1
- n2 is dimension 2

For 2 dimensional grids, band packing of 0 is referred to as band sequential, 1 line interleaved and 2 pixel interleaved.

**Type:** Integer {frozen}
3.1.7.2 GC_GridGeometry

The GC_GridGeometry data type describes the valid range of grid coordinates and the math transform to transform grid coordinates to real world coordinates. This data type is shown in figure 17.

![Figure 17: GC_GridGeometry data type.](image)

3.1.7.2.1 Attributes

3.1.7.2.1.1 +gridRange : GC_GridRange

**Description:** The valid coordinate range of a grid coverage. The lowest valid grid coordinate is zero.

A grid with 512 cells can have a minimum coordinate of 0 and maximum of 512, with 511 as the highest valid index.

**Type:** GC_GridRange \{frozen\}

3.1.7.2.1.2 +gridToCoordinateSystem : CT_MathTransform

**Description:** The CT_MathTransform allows for the transformations from grid coordinates to real world earth coordinates. The transform is often an affine transformation. The coordinate system of the real world coordinates is given by the coordinateSystem : CS_CoordinateSystem attribute on the CV_Coverage.

Note: If no math transform is given, gridToCoordinateSystem will be NULL.

**Type:** CT_MathTransform \{frozen\}

3.1.7.3 GC_GridRange

GC_GridRange defines a range of grid coverage coordinates. For example this data type is used to access a block of grid coverage data values. This is shown in figure 18.

![Figure 18: GC_GridRange data type.](image)
3.1.7.3.1 Attributes

3.1.7.3.1.1 +lo : Sequence<Integer>

**Description:** The valid minimum inclusive grid coordinate. The sequence contains a minimum value for each dimension of the grid coverage. The lowest valid grid coordinate is zero.

**Type:** Sequence<Integer> {frozen}

3.1.7.3.1.2 +hi : Sequence<Integer>

**Description:** The valid maximum exclusive grid coordinate. The sequence contains a maximum value for each dimension of the grid coverage.

**Type:** Sequence<Integer> {frozen}

3.1.7.4 GC_ParameterInfo

This structure contains information about a parameter required for a grid coverage processing operation. This information includes the name of the parameter, parameter description, parameter type etc. This data type is illustrated in figure 19.

![GC_ParameterInfo data type](figure)

**Figure 19:** GC_ParameterInfo data type.

3.1.7.4.1 Attributes

3.1.7.4.1.1 +name : CharacterString

**Description:** Parameter name.

**Type:** CharacterString {frozen}

3.1.7.4.1.2 +description : CharacterString

**Description:** Parameter description. If no description, the value will be an empty string.

**Type:** CharacterString {frozen}

3.1.7.4.1.3 +type : GC_ParameterInfoType

**Description:** Parameter type. The enumeration contains standard parameter types for integer, string, floating-point numbers, objects, etc.

**Type:** GC_ParameterInfoType {frozen}
3.1.7.4.1.4 (+defaultValue : Object)

**Description:** Default value for parameter. The type Object can be any type including a Number or a CharacterString. For example, a filtering operation could have a default kernel size of 3.

Note: If there is no default value, defaultValue will be empty.

**Type:** Object {frozen}

3.1.7.4.1.5 (+minimumValue : Number)

**Description:** Minimum parameter value. For example, a filtering operation could have a minimum kernel size of 3.

Note: If there is no minimum value, minimumValue will be empty.

**Type:** Number {frozen}

3.1.7.4.1.6 (+maximumValue : Number)

**Description:** Maximum parameter value. For example, a filtering operation could have a maximum kernel size of 9.

Note: If there is no maximum value, maximumValue will be empty.

**Type:** Number {frozen}

3.1.7.5 GC_Parameter

The parameter codelist required for a grid coverage processing operation. This structure contains the parameter name (as defined from the GC_ParameterInfo structure) and it’s value. This data type is shown in figure 20.

![Figure 20: GC_Parameter data type.](image)

3.1.7.5.1 Attributes

3.1.7.5.1.1 (+name : CharacterString)

**Description:** Parameter name.

**Type:** CharacterString {frozen}

3.1.7.5.1.2 (+value : Object)

**Description:** The value for parameter. The type Object can be any type including a Number, a CharacterString or an instance of an interface. For example, a grid processor operation will typically require a parameter for the input grid coverage. This parameter may have “Source” as the parameter name and the instance of the grid coverage as the value.
3.1.7.6 GC_Format

This interface is a discovery mechanism for the information on each format supported by the GC_GridCoverageExchange. This interface is shown in figure 21.

| + name : CharacterString |
| + description : CharacterString |
| + vendor : CharacterString |
| + docURL : CharacterString |
| + version : CharacterString |
| + numParameters : Integer |
| + getParameterInfo(index : Integer) : GC_ParameterInfo |

Figure 21: Context Diagram: GC_Format.

3.1.7.6.1 Attributes

3.1.7.6.1.1 +name : CharacterString

**Description:** Name of the file format. This name is used as the name of the file in the exportTo operation.

**Type:** CharacterString {frozen}

3.1.7.6.1.2 +description : CharacterString

**Description:** Description of the file format. If no description, the value will be an empty string.

**Type:** CharacterString {frozen}

3.1.7.6.1.3 +vendor : CharacterString

**Description:** Vendor or agency for the format.

**Type:** CharacterString {frozen}

3.1.7.6.1.4 +docURL : CharacterString

**Description:** Documentation URL for the format.

**Type:** CharacterString {frozen}

3.1.7.6.1.5 +version : CharacterString

**Description:** Version number of the format.

**Type:** CharacterString {frozen}
### 3.1.7.6.1.6 +numParameters : Integer

**Description:** Number of optional parameters for the exportTo operation.

**Type:** Integer \{frozen\}

### 3.1.7.6.2 getParameterInfo

**Prototype:** getParameterInfo(index : Integer) : GC_ParameterInfo

**Class:** GC_Format

**Description:** Retrieve the parameter information for a given index.

**Inputs:**

- **index**: Index to the parameter.

**Outputs:** None

**Returns:** GC_ParameterInfo

**Exceptions**

- GC_InvalidIndex

### 3.1.7.7 GC_ValueInBytePacking

Order of the pixels in a byte for CV_1BIT, CV_2BIT and CV_4BIT grid values. This codelist is shown in figure 22.

```plaintext
<<CodeList>>
GC_ValueInBytePacking
+ GC_LoBitFirst : Integer = 0
+ GC_HiBitFirst : Integer = 1
```

**Figure 22: Context Diagram: GC_ValueInBytePacking**

### 3.1.7.8 GC_ByteInValuePacking

Specifies the order of the bytes in multi-byte values. GC_wkbXDR is Big Endian. GC_wkbNDR is Little Endian. This is shown in figure 23.

```plaintext
<<CodeList>>
GC_ByteInValuePacking
+ GC_wkbXDR : Integer = 0
+ GC_wkbNDR : Integer = 1
```

**Figure 23: Context Diagram: GC_ByteInValuePacking**

### 3.1.7.9 GC_ParameterType

Specifies the type of a parameter value. A sequence of parameters is used to pass a variable number of arguments to an operation and each of these parameters can have a different parameter type. An operation requiring a string and grid coverage would use two parameters for type GC_StringType and GC_GridCoverageType. This codelist is shown in figure 24.
3.1.8 Supporting Interfaces in Grid Coverage Processing (GP) Package

This interface provides different methods to access grid coverage values and image-processing capabilities for grid coverages. These processors will transform values within a single sample dimension, and leave the values in other sample dimensions unaffected. The modified sample dimension may also change its type (e.g. from CV_4BIT to CV_1BIT). The actual underlying grid data remains unchanged.

3.1.8.1 GP_Operation

This interface provides descriptive information for a grid coverage processing operation. The descriptive information includes such information as the name of the operation, operation description, and number of source grid coverages required for the operation. The operations and attributes of this interface is given figure 25.
3.1.8.1.1 Attributes

3.1.8.1.1.1 +name : CharacterString

**Description:** Name of the processing operation.
**Type:** CharacterString {frozen}

3.1.8.1.1.2 +description : CharacterString

**Description:** Description of the processing operation. If no description, the value will be an empty string.
**Type:** CharacterString {frozen}

3.1.8.1.1.3 +vendor : CharacterString

**Description:** Implementation vendor name.
**Type:** CharacterString {frozen}

3.1.8.1.1.4 +docURL : String

**Description:** URL for documentation on the processing operation. If no online documentation is available the string will be empty.
**Type:** String {frozen}

3.1.8.1.1.5 +version : String

**Description:** Version number for the implementation.
**Type:** String {frozen}

3.1.8.1.1.6 +numSources : Integer
Description: Number of source grid coverages required for the operation.
Type: Integer {frozen}

3.1.8.1.1.7 +numParameters : Integer
Description: Number of parameters for the operation.
Type: Integer {frozen}

3.1.8.1.2 getParameterInfo
Prototype: getParameterInfo(index : Integer) : GC_ParameterInfo
Class: GP_Operation
Description: Retrieve the parameter information for a given index.
Inputs:
index Parameter information index to retrieve. Index starts at 0.
Outputs: None
Returns: GC_ParameterInfo
Exceptions: GP_InvalidIndex

3.2 Well-known Binary Representations

3.2.1 WKBGeoTIFF

GeoTIFF represents an effort by over 160 different remote sensing, GIS, cartographic, and surveying related companies and organizations to establish a TIFF based interchange format for georeferenced raster imagery.

The initial efforts to define a TIFF "geotie" specification began under the leadership of Ed Grissom at Intergraph, and others in the early 1990's. In 1994 a formal GeoTIFF mailing-list was created and maintained by Niles Ritter at JPL which quickly grew to over 140 subscribers from government and industry. The purpose of the list is to discuss common goals and interests in developing an industry-wide GeoTIFF standard. This culminated in a conference in March of 1995 hosted by SPOT Image, with representatives from USGS, Intergraph, ESRI, ERDAS, SoftDesk, MapInfo, NASA/JPL, and others, in which the current working proposal for GeoTIFF was outlined. Niles Ritter, and Mike Ruth of SPOT Image condensed the outline into a prerelease GeoTIFF specification document. Following discussions with Dr. Roger Lott of the European Petroleum Survey Group (EPSG), the GeoTIFF projection parameterization method was extensively modified, and brought into compatibility with both the POSC Epicentre model, and the Federal Geographic Data Committee (FGDC) metadata approaches.

GeoTIFF fully complies with the TIFF 6.0 specifications. Its extensions do not in any way go against the TIFF recommendations, nor do they limit the scope of raster data supported by TIFF.

GeoTIFF uses a small set of reserved TIFF tags to store a broad range of georeferencing information, catering to geographic as well as projected coordinate systems needs. Projections include UTM, US State Plane and National Grids, as well as the underlying projection types such as Transverse Mercator, Lambert Conformal Conic, etc. No information is stored in private structures, IFD's or other mechanisms which would hide information from naive TIFF reading software.

More information on GeoTIFF can be found at:

GeoTIFF Home page:
3.3 GP_GridCoverageProcessor Operations

The following are examples of grid coverage processing operations. This list is not an exhaustive list. Different GP_GridCoverageProcessor implementations will provide different operations and need not implement the entire list below.

Using one of these operations to change the way the grid is being accessed will not affect the state of the grid coverage controlled by another operations. For example, changing the interpolation method should not affect the number of sample dimensions currently being accessed or value sequence.

The GP_Operation tables below contains the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Name of the processing operation. This name is passed as a parameter to the GP_GridCoverageProcessor DoOperation method to instantiate a new grid coverage on which the processing operation is performed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Description of the processing operation.</td>
</tr>
<tr>
<td>Vendor</td>
<td>Vendor of the processing operation.</td>
</tr>
<tr>
<td>DocURL</td>
<td>Documentation URL for the processing operation. This value will be an empty string (&quot;&quot;) if no documentation URL is provided.</td>
</tr>
<tr>
<td>Version</td>
<td>Version number of the implementation.</td>
</tr>
<tr>
<td>Number Source</td>
<td>Number of source grid coverages required for the operation.</td>
</tr>
<tr>
<td>Number Parameters</td>
<td>Number parameters required for the operation. The GC_ParameterInfo table contains this number of rows.</td>
</tr>
</tbody>
</table>

The GC_ParameterInfo tables below contains one the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Name of parameter. The parameter name is used in the GC_Parameter name value structure.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Parameter description</td>
</tr>
</tbody>
</table>
GC_ParameterType Parameter type see appendix A.2.2. Enumerations Supporting GC_GridCoverage for details.

Default Value Default value of the parameter. No default value will be given (indicated by NA) if it is inappropriate for the parameter type. Those entries in the table with an * prefix indicates an implied default.

Minimum Value Minimum value of the parameter. No default value will be given (indicated by NA) if it is inappropriate for the parameter type.

Maximum Value Maximum value of the parameter. No default value will be given (indicated by NA) if it is inappropriate for the parameter type.

3.3.1 Gray Scale Threshold

A gray scale threshold classifies the grid coverage values into a Boolean value. This sample dimension will be modified into a Boolean value and the dimension type of the source sample dimension will be represented as CV_1BIT.

**GP_Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Threshold”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

**GC_ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
<tr>
<td>“Threshold”</td>
<td>Vendor Specific</td>
<td>GC_NumberType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Source Source grid coverage to apply gray scale threshold.

SampleDimension Sample dimension on which the operation will be applied. This sample dimension will be modified into a Boolean value and the dimension type of the source sample dimension will be CV_1BIT.

Threshold Threshold value. Values below the threshold will be given a value of zero and values above or equal to will be given a one. A “no data value” will be given a zero value.

Exceptions GP_InvalidSampleDimension GP_InvalidThresholdValue

3.3.2 Image Enhancements

Image enhancements are used to improve the contrast of grid data values in sample dimensions which have a small range of data values. For example, a sample dimension with an 8 bit unsigned dimension type may only have a data range from 0 to 100. This sample dimension will appear dark when visualized. To improve the visualization of the grid data values, a transformation or enhancement can be applied to the grid values to transform the values to a range
from 0 to 255 thus improving the visualization of the grid data. The modified sample dimension will have a sample dimension type of CV_8BIT_U.

3.3.2.1 Linear Enhancement

Perform a linear enhancement on a sample dimension of a grid coverage. The range of grid values is expanded uniformly to fill the range of 0 to 255. Close grid values may now be distinguished when visualized.

A linear enhancement uses the formula:

$$\text{New value} = \frac{\text{Value} - \text{Min}}{\text{Max} - \text{Min}} \times 255$$

Where
- **New value** is the 8-bit unsigned output grid value
- **Value** is the input grid value
- **Min** is the minimum grid value in the sample dimension
- **Max** is the maximum grid value in the sample dimension

Note: When converting the floating-point new value to 8-bit integer, the new value is rounded.

### GP_Operation

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Source</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>“LinearEnhancement”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### GC_ParameterInfo

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
</tbody>
</table>

**Source**

Source grid coverage to apply enhancement.

**SampleDimension**

Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will be changed to CV_8BIT_U.

**Exceptions**

GP_InvalidSampleDimension

3.3.2.2 Root Enhancement

Perform a root enhancement on a grid coverage sample dimension.

A root enhancement uses the formula:

$$\text{New value} = \sqrt{\frac{\text{Value} - \text{Min}}{\text{Max} - \text{Min}}} \times 255$$

Where
- **New value** is the 8-bit unsigned output grid value
- **Value** is the input grid value
- **Min** is the minimum grid value in the sample dimension
Max  maximum grid value in the sample dimension

Note: When converting the floating-point new value to 8-bit integer, the new value is rounded.

**GP Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>“RootEnhancement”</td>
<td></td>
<td>Vendor</td>
<td>Vendor</td>
<td>Vendor</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**GC_ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
</tbody>
</table>

Source  Source grid coverage to apply enhancement.

SampleDimension  Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will be changed to CV_8BIT_U.

Exceptions  GP_InvalidSampleDimension

**3.3.2.3 Equalization Enhancement**

Perform an equalization enhancement on a grid coverage sample dimension. Grid values are assigned new values based on their frequency of occurrence. A greater range of new values is assigned to portions of the histogram with the higher frequency of occurrence.

**GP Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>“EqualizationEnhancement”</td>
<td></td>
<td>Vendor</td>
<td>Vendor</td>
<td>Vendor</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**GC_ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
</tbody>
</table>

Source  Source grid coverage to apply enhancement.

SampleDimension  Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will be changed to CV_8BIT_U.
Exceptions GP_InvalidSampleDimension

3.3.2.4 Infrequency Enhancement

Perform a infrequency enhancement on a grid coverage sample dimension. New grid values are produced where the infrequently occurring values in the input image are mapped to 255 while the frequently occurring values are mapped to 0.

**GP_Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>“InfrequencyEnhancement”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**GC_ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
</tbody>
</table>

Source Source grid coverage apply enhancement.

SampleDimension Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will be changed to CV_8BIT_U.

Exceptions GP_InvalidSampleDimension

3.3.3 Interpolate

This operation specifies the interpolation type to be used to interpolate values for points which fall between grid cells. The default value is nearest neighbor. The new interpolation type operates on all sample dimensions.

**GP_Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Interpolate”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**GC_ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“Type”</td>
<td>Vendor Specific</td>
<td>GC_StringType</td>
<td>“NearestNeighbor”</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>
Source grid coverage.

Possible values for type:

- **NearestNeighbor**
  
  For a given point \( p \), find the grid cell closest to \( p \) and return the values of that grid cell.

- **Bilinear**
  
  For a given point \( p \), suppose \( p \) is contained between grid cells \( C_1, C_2, C_3 \) and \( C_4 \) with values \( W_1, W_2, W_3 \) and \( W_4 \). The location of point \( p \) can be defined at a location \( i, j \) where \( 0 \leq i \leq 1 \) and \( 0 \leq j \leq 1 \) between these cells. The value \( W \) returned will be determined by:
  
  \[
  W = (1 - i)(1 - j)W_1 + i(1 - j)W_2 + (1 - i)jW_3 + ijW_4
  \]

  If grid value has an integer sample dimension type, \( W \) will be rounded to the nearest integer value.

- **Optimal**
  
  The optimum interpolation technique may be calculated using Wiener filters and depends on factors that include the spectral power distribution of the scene, the sampling frequency and spectral power distribution of the noise in the image. (See: Rosenfeld, A. and A. C. Kak, *Digital Picture Processing, Second Ed, Vol 1*, Academic Press, 1982).

  If grid value has an integer sample dimension type, \( W \) will be rounded to the nearest integer value.

- **Bicubic**
  
  Cubic polynomials may be used to approximate optimal interpolation filters. (See: Rosenfeld, A. and A. C. Kak, *Digital Picture Processing, Second Ed, Vol 1*, Academic Press, 1982).

  If grid value has an integer sample dimension type, \( W \) will be rounded to the nearest integer value.

**Exceptions**

- GP_InvalidInterpolationType

### 3.3.4 Band Ratioing

Perform a band ratio operation. This operation will calculate \((\text{value of sample dimension 1})/(\text{value of sample dimension 2})\). The resulting new value will be zero if: 1) the value of sample dimension 2 is zero or 2) either sample dimension 1 or sample dimension 2 have the “no data value”. This operation will modify sample dimension 1 (sample dimension 1 will contain the output values). The sample dimension type of the modified sample dimension will be `CV_32BIT_REAL`.

**GP Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>“BandRatio”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

**GC_ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension1”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on</td>
</tr>
</tbody>
</table>
“SampleDimension2”  Vendor Specific  GC_IntegerType  1  0  
Grid coverage processed
Dependent on
Grid coverage processed

Source  Source grid coverage.

SampleDimension1  Source sample dimension 1 to perform band ratio. This sample dimension will contain the modified grid data values.

SampleDimension2  Source sample dimension 2 to perform band ratio.

Exceptions  GP_InvalidSampleDimension

3.3.5 Spatial Filtering

Filtering is an enhancement operation that alters the grid values on the basis of the neighborhood grid values. For this reason, filtering is considered to be a spatial or area operation. There are many different filters that can be applied to a grid coverage but the general concept of filtering is the same. A filter window or kernel is defined, its dimension being an odd number in the x and y dimensions. Each cell in this window contains a co-efficient or weighting factor representative of some mathematical relationship. A filtered grid coverage is generated by multiplying each co-efficient in the window by the grid value in the original grid coverage corresponding to the window’s current location and assigning the result to the central pixel location of the window in the filtered grid coverage. The window is moved throughout the grid coverage on pixel at a time. This window multiplication process is known as convolution.

A grid coverage contains both low and high spatial information. High frequencies describe rapid change from one grid cell to another such as roads or other boundary conditions. Low frequencies describe gradual change over a large number of cells such as water bodies. High pass filters allow only high frequency information to be generated in the new grid coverage Grid coverages generated with high pass filters will show edge conditions. Low pass filters allow low frequency information to be generated in the new grid coverage. Examples of both low and high pass filters are given below.

The grid coverage produced from a filtering operation will have the same dimension as the source grid coverage. To produce filtered values around the edges of the source grid coverage, edge rows and columns will be duplicated to fill a complete kernel.

3.3.5.1 Mean Filter

Perform a mean filter operation on a grid coverage. The new grid value will be the average of all the grid values in kernel. This filter results in a smoothing of the data thus eliminating noise.

New grid value = \( \frac{\text{sum } (V(i, j))}{(Xsize + YSize)} \)

Where  
\( i \)  x position in the kernel  
\( j \)  y position in the kernel  
\( Xsize \)  X size of the kernel  
\( Ysize \)  Y Size of the kernel  
\( V(i, j) \)  value at position i, j
### 3.3.5.2 Mode Filter

Perform a mode filter operation on a grid coverage. The new grid value is given the most frequently occurring value in the kernel. This filter is primarily used to cleanup thematic grid data since this filter replaces small island cells by their large surrounding grid data values.

#### GP Operation

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ModeFilter”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

#### GC_ParameterInfo

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
<tr>
<td>“Xsize”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>3</td>
<td>1</td>
<td>Vendor Specific</td>
</tr>
<tr>
<td>“Ysize”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>3</td>
<td>1</td>
<td>Vendor Specific</td>
</tr>
</tbody>
</table>

Source: Source grid coverage.
SampleDimension: Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will not be changed.
XSize: Number of rows for the filter kernel.
YSize: Number of columns for the filter kernel.

Exceptions: GP_InvalidSampleDimension, FilterSizeMustBeOdd
3.3.5.3 Median Filter

Perform a median filter operation on a grid coverage. The new grid value will be the median of all the grid values in the kernel. This filter results in a smoothing of the grid values.

**GP_Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>“MedianFilter”*</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

**GC_ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Source grid coverage.</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will not be changed</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
<tr>
<td>“Xsize”</td>
<td>Number of rows for the filter kernel.</td>
<td>GC_IntegerType</td>
<td>3</td>
<td>1</td>
<td>Vendor Specific</td>
</tr>
<tr>
<td>“Ysize”</td>
<td>Number of columns for the filter kernel.</td>
<td>GC_IntegerType</td>
<td>3</td>
<td>1</td>
<td>Vendor Specific</td>
</tr>
</tbody>
</table>
3.3.5.4 Gaussian Filter

Perform a gaussian filter operation on a grid coverage. This filter is a low pass filter to blur the grid coverage. This filter uses the following Gaussian function to compute the filter weights:

\[ G(i, j) = \exp\left(-\frac{(i - u)^2 + (j - v)^2}{2 \times SIGMSQ}\right) \]

Where

- \(i, j\) the cell within the kernel
- \(u, v\) the center of the kernel
- \(SIGMSQ\) set to 4

The filter weights \(W(i, j)\) are normalized values of \(G(i, j)\) over the entire kernel, hence the sum of all weights is 1.

The filtered grid values is the sum of \(W(i, j) \times V(i, j)\) over all the grid values in the kernel, where \(V(i, j)\) is the original grid value at location \((i, j)\).

**GP_Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>“GausianFilter”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

**GC_ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>DefaultValue</th>
<th>MinimumValue</th>
<th>MaximumValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed Vendor Specific</td>
</tr>
<tr>
<td>“Xsize”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>3</td>
<td>1</td>
<td>Vendor Specific</td>
</tr>
<tr>
<td>“Ysize”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>3</td>
<td>1</td>
<td>Vendor Specific</td>
</tr>
</tbody>
</table>

Source

Source grid coverage.

SampleDimension

Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will not be changed.

XSize

Number of rows for the filter kernel.

YSize

Number of columns for the filter kernel.

Exceptions

- GP_InvalidSampleDimension
- GP_FilterSizeMustBeOdd
3.3.5.5 LaplacianType 1 Filter

Perform a laplacian filter operation on a grid coverage. This is a high pass filter which highlights the edges having positive and negative brightness slopes. This filter multiples the co-efficients in figure 3.4.12 with the corresponding grid data value in the kernel window.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3.4.12  Kernel co-efficients for Laplacian Type 2 Filter

The new grid value will be calculated as the sum of (grid value * co-efficient) for each kernel cell divided by 9.

GP Operation

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>“LaplacianType1Filter”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

GC_ParameterInfo

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
</tbody>
</table>

Source  Source grid coverage.

SampleDimension  Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will not be changed

Exceptions  GP_InvalidSampleDimension

3.3.5.6 Laplacian Type 2 Filter

Perform a laplacian filter operation on a grid coverage. This is a high pass filter which highlights the edges having positive and negative brightness slopes. This filter multiples the co-efficients in figure 3.4.13 with the corresponding grid data value in the kernel window.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>8</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Figure 3.4.13  Kernel co-efficients for Laplacian Type 2 Filter
The new grid value will be calculated as the sum of (grid value * co-efficient) for each kernel cell divided by 9.

**GP Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>“LaplacianType2Filter”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**GC ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimension”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>0</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
</tbody>
</table>

Source  Source grid coverage.

SampleDimension  Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will not be changed

Exceptions  GP_InvalidSampleDimension

### 3.3.6 Select Sample Dimensions

This operation returns an GC_GridCoverage which uses a subset of the sample dimensions from an existing GC_GridCoverage. For example, if sample dimensions 5, 9 and 10, then the new GC_GridCoverage will contain three sample dimensions numbered 0, 1 and 2.

This operation can also be used to “clone” sample dimensions, by repeating sample dimension index values. This could be useful if an application intends to overlay an adapted sample dimension with its original unadapted version.

**GP Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>“SelectSampleDimension”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**GC ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“SampleDimensions”</td>
<td>Vendor Specific</td>
<td>GC_VectorType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Source  Source grid coverage.
SampleDimensions: Sample dimensions on which the operation will be applied. The dimension type of the source sample dimensions will not be changed.

Exceptions: GP_InvalidSampleDimension

### 3.3.7 Resample

Resample a grid coverage using a different grid geometry. This interface provides the following functionality:

**Resampling**

The grid coverage can be resampled at a different cell resolution. Some implementations may be able to do resampling efficiently at any resolution. This can be determined from the GP_GridCoverageProcessor metadata HasArbitraryResolutions keyword. Also a non-rectilinear grid coverage can be accessed as rectilinear grid coverage with this interface.

**Reprojecting**

The new grid geometry can have a different coordinate system than the underlying grid geometry. For example, a grid coverage can be reprojected from a geodetic coordinate system to Universal Transverse Mercator coordinate system.

**Subsetting**

A subset of a grid can be viewed as a separate coverage by using this operation with a grid geometry which as the same geotferencing and a region. Grid range in the grid geometry defines the region to subset in the grid coverage.

### GP Operation

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Resample”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

### GC_ParameterInfo

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimu m Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“InterpolationType”</td>
<td>Vendor Specific</td>
<td>GC_StringType</td>
<td>“NearestNieghbor”</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“CoordinateSystem”</td>
<td>Vendor Specific</td>
<td>CS_CoordinateSystemType</td>
<td>*Same as the source grid coverage</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“GridGeometry”</td>
<td>Vendor Specific</td>
<td>GC_GridGeometryType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

**Source**: Source grid coverage.

**SampleDimension**: Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will not be changed.

**InterpolationType**: Interpolation type to use when a grid is being resampled to a lower resolution and the resampled resolution is a multiple of the grid resolution.
Recognized interpolation types are:

- “NearestNeighor” Use the value of the nearest cell.
- “BlockAverage” The resampled cell value is the average all the cells.
- “BlockMode” The resampled cell value is the value of the cell which occurs most frequent.

Other interpolation types are implementation specific.

CoordinateSystem Coordinate system to which the grid coverage is reprojected. If the operation only does a subsetting of the grid coverage this parameter is omitted.

GridGeometry Grid geometry to resample grid coverage.

Exceptions GP_CannotReproject GP_InvalidInterpolationType

3.3.8 Sequence

Returns an GC_GridCoverage which uses a different traversal order for the grid values.

### GP Operation

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Sequence”</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

### GC_ParameterInfo

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Source”</td>
<td>Vendor Specific</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“BandPacking”</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>*Same as source</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“ValueInBytePacking”</td>
<td>Vendor Specific</td>
<td>GC_ValueInBytePackingType</td>
<td>*Same as source</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>“ByteInValuePacking”</td>
<td>Vendor Specific</td>
<td>GC_ByteInValuePackingType</td>
<td>*Same as source</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Source Source grid coverage.

BandPacking Order in which to traverse the values in the grid.

ValueInBytePacking Order of the bits within the byte for value using less than 8 bits.

ByteInValuePacking Order of the bytes in a grid value for values with sample dimensions greater than 8 bits.

Exceptions None
3.3.9 Density Slice

Performs a density slice operation on a grid coverage. If the number of breakpoints specified is n, then the grid coverage source sample dimension will be classified into n + 2 values. The color table must have the same number of color entries as the number of new values. The ranges of the new categories may be retrieved from the category names in the CV_SampleDimension interface.

For example, a density slice on an 8-bit sample dimension using breakpoints: 50, 100, 150 will give the following categories:

<table>
<thead>
<tr>
<th>Range</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>No data</td>
<td>0</td>
</tr>
<tr>
<td>&lt;= 50</td>
<td>1</td>
</tr>
<tr>
<td>50 – 100</td>
<td>2</td>
</tr>
<tr>
<td>100 – 150</td>
<td>3</td>
</tr>
<tr>
<td>150 &lt;</td>
<td>4</td>
</tr>
</tbody>
</table>

**GP Operation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Vendor</th>
<th>Doc URL</th>
<th>Version</th>
<th>Number Source</th>
<th>Number Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DensitySlice&quot;</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>Vendor Specific</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

**GC_ParameterInfo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>GC_ParameterType</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Source&quot;</td>
<td>&quot;Grid Coverage&quot;</td>
<td>GC_GridCoverageType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>&quot;SampleDimension&quot;</td>
<td>Vendor Specific</td>
<td>GC_IntegerType</td>
<td>NA</td>
<td>0</td>
<td>Dependent on Grid coverage processed</td>
</tr>
<tr>
<td>&quot;BreakPoints&quot;</td>
<td>&quot;Break points&quot;</td>
<td>GC_VectorType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>&quot;ColorPalette&quot;</td>
<td>&quot;Color Palette&quot;</td>
<td>GC_VectorType</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Source grid coverage.

SampleDimension Sample dimension on which the operation will be applied. The dimension type of the source sample dimension will be changed to CV_8BIT_U

BreakPoints Break points to classify the grid values.

ColorPalette Color palette associated with the classified grid.

Exceptions GP_InvalidSampleDimension GP_ColorPaletteMismatch

4 REFERENCES

1. The Open GIS Specification Model Topic 6: The Coverage Type and Its Subtypes Version 3.2
2. OpenGIS Simple Features Specification For OLE/COM Revision 1.0
3. OpenGIS Simple Features Specification For CORBA Revision 1.0
4. OpenGIS Simple Features Specification For SQL Revision 1.0
APPENDICES

A.1 GLOSSARY OF TERMS

Sample dimension Sample dimension relates to particular values occurring at a spatial position. For a point coverage with three values at every point, has three sample dimensions and which are often referred to as “fields”. Sample dimensions are referred to as “bands” for a grid coverage.

Overview A pre-calculated lower resolution representation of a grid coverage. A grid coverage can have a number of overviews. For example, a grid coverage with 1 metre pixel resolution may have overviews for 3, 9, and 27 metre resolutions. Overview resolution need not be a multiple resolution of the full data set. An overview is typically used to enhance display performance for large grid coverages.

A.2 SUPPORTING WELL-KNOWN VALUES

The following are Well-known values specific to this submission.

<table>
<thead>
<tr>
<th>CV_PaletteInterpretation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gray</td>
<td>C1: Gray [0..255], with Black=0 and White=255.</td>
</tr>
<tr>
<td></td>
<td>C2: Unused [0]</td>
</tr>
<tr>
<td></td>
<td>C3: Unused [0]</td>
</tr>
<tr>
<td></td>
<td>C4: Unused [0]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CV_PaletteInterpretation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>C1: Red [0..255]</td>
</tr>
<tr>
<td></td>
<td>C2: Green [0..255]</td>
</tr>
<tr>
<td></td>
<td>C3: Blue [0..255]</td>
</tr>
<tr>
<td></td>
<td>C4: Alpha[0..255] (0 if Alpha not present)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CV_PaletteInterpretation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMYK</td>
<td>C1: Cyan [0..255]</td>
</tr>
<tr>
<td></td>
<td>C2: Magenta [0..255]</td>
</tr>
<tr>
<td></td>
<td>C3: Yellow [0..255]</td>
</tr>
<tr>
<td></td>
<td>C4: Black[0..255]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CV_PaletteInterpretation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSL</td>
<td>C1: Hue [0..360]</td>
</tr>
<tr>
<td></td>
<td>C2: Saturation [0..100]</td>
</tr>
<tr>
<td></td>
<td>C3: Lightness [0..100]</td>
</tr>
<tr>
<td></td>
<td>C4: Unused[0]</td>
</tr>
</tbody>
</table>
A.3 EXCEPTIONS, ERRORS AND ERROR CODES

For the COM interfaces, exceptions will be returned using the HRESULTS of the interfaces. All interfaces will return a non-negative value (e.g. S_OK) on success or a negative value (e.g. E_FAIL) on failure.

CORBA error codes will be reported as exceptions.

Any error codes not listed below will indicate a provider specific error has occurred.

Specific Exceptions

CV_InvalidIndex
CV_MetadataNameNotFound
CV_PointOutsideCoverage

GC_InvalidIndex
GC_InvalidRange
GC_GridNotEditable
GC_MetadataNameNotFound
GC_CannotCreateGridCoverage
GC_FileFormatNotCompatiableWithGridCoverage
GC_ErrorExportingGridCoverage
GC_InvalidParameterName
GC_InvalidParameterValue

GP_InvalidSampleDimension
GP_GridCoverageHasOnlyOneSampleDimension
GP_MetadataNameNotFound
GP_InvalidIndex
GP_OperationNotFound
GP_InvalidParameterName
GP_InvalidParameterValue
GP_InvalidThresholdValue
GP_InvalidInterpolationType
GP_FilterSizeMustBeOdd
GP_CannotReproject
GP_ColorPaletteMismatch

Description: Specifies the different exceptions for the implementation specification.

A.4 CONVERSION RULES FOR GRID VALUES TO SPECIFIC DATA TYPES

Conversion of Value to Boolean

If value is 0 then Boolean is FALSE else Boolean is TRUE.

Conversion of Value to 2 bit integer

For real values, value is truncated to integer (rounded down). If value is less than 0, the value is zero. If value is greater than 3, the character value is 3.

Conversion of Value to 4 bit integer

For real values, value is truncated to integer (rounded down).
If value is less than 0, the value is zero.
If value is greater than 15, the character value is 15.

**Conversion of Value to Unsigned Character (0 to 255)**

For real values, value is truncated to integer (rounded down).
If value is less than 0, the unsigned character value is zero.
If value is greater than 255, the character value is 255

**Conversion of Value to Signed Character (-128 to 127)**

For real values, value is truncated to integer (rounded down).
If value is less than -128, the signed character value is -128.
If value is greater than 127, the signed character value is 127.

**Conversion of Value to Unsigned Short (0 to 65535)**

For real values, value is truncated to integer (rounded down).
If value is less 0, unsigned short is 0.
If value is greater than 65535, unsigned short is 65535.

**Conversion of Value to Signed Short (-32768 to 32767)**

For real values, value is truncated to integer (rounded down).
If value is less –32768, signed short is -32768.
If value is greater than 32767, signed short is 32767

**Conversion of Value to Unsigned Integer (0 to 0xffffffff)**

For real values, value is truncated to integer (rounded down).
If value is less 0, unsigned integer is 0.
If value is greater than 0xffffffff, unsigned integer is 0xffffffff.

**Conversion of Value to Signed Integer (-2147483648 to 2147483647)**

For real values, value is truncated to integer (rounded down).
If value is less -2147483648, signed integer is -2147483648.
If value is greater than 2147483647, signed integer is 2147483647.

**Conversion of Value to Float**

No conversion rules necessary

**Conversion of Value to Double**

No conversion rules necessary

### A.5 COM IDL SPECIFICATION

The detailed interface specification is defined using Microsoft’s Interface Definition Language (MIDL).

This specification consists of three MIDL files: OGC.CV.IDL (Coverage package), OGC_GC.IDL (Grid Coverage package) and OGC_GP.IDL (Grid Coverage Processing package).
The MIDL files require three Coordinate Transformation MIDL files (OGC_PT.IDL, OGC_CS.IDL and OGC_CT.IDL).

A.6 CORBA IDL SPECIFICATION

The detailed interface specification is defined using CORBA Interface Definition Language (IDL).

This specification is consists of three IDL files: OGC_CV.IDL (Coverage package), OGC_GC.IDL (Grid Coverage package) and OGC_GP.IDL (Grid Coverage Processing package).

The IDL files require three Coordinate Transformation IDL files (OGC_PT.IDL, OGC_CS.IDL and OGC_CT.IDL).

A.7 SPECIFICATION SEQUENCE DIAGRAMS

A.7.1 SIMPLE GRID COVERAGE CREATION EXAMPLE

Figure 26 illustrates the creation a grid coverage from a grid coverage exchange interface and how to perform a simple grid processing operation (“Threshold”). Grid values are retrieved from the new grid. The grid coverage processor will fetch grid values from the source grid coverage (grid) as required.

![Figure 26: Simple grid coverage creation](image-url)
A.7.2 GRID COVERAGE PROCESSING USING THE DISCOVERY MECHANISM

Figure 27 illustrates the creation a grid coverage from a grid coverage exchange interface and invoking a processing operation after discovering the available operations. Grid values are retrieved from the modified grid coverage (newgrid) and the grid coverage processor fetches grid values from the source grid coverage as required.

![Diagram of grid coverage processing using the discovery mechanism]

A.8 VISUAL BASIC SAMPLE CODE

A.8.1 READING GRID DATA VALUES

The following pseudo code illustrates creation of a grid coverage (Grid) from a grid coverage exchange (GridExchange) and retrieving grid values. The grid coverage is created from a GeoTIFF file: “test.tif”. The grid values are retrieved as bytes for the full image since the geometry of the grid coverage is used as input to DataBlockAsByte.

```vbnet
Public GridExchange As GC_GridCoverageExchange
Dim Grid As GC_GridCoverage
Dim GridGeometry As GC_GridGeometry
```
Dim Buf() As Byte

Rem create an instance of a grid coverage exchange
Set GridExchange = New PCI_GC_GridCoverageExchange

Rem create the grid coverage from a GeoTIFF file
Set Grid = GridExchange.CreateFromName(“test.tif”)

Rem fetch the grid geometry for the GeoTIFF grid coverage
GridGeometry = Grid/GridGeometry

Rem fetch the grid values for the full grid coverage
Buf = Grid/DataBlockAsByte(GridGeometry/GridRange)