International
Virtual
Observatory
Alliance

IVOA N-Dimensional Cube Model

Version 1.0

IVOA Working Draft 20170203

This version:

WD-CubeDM-1.0-20170203

Previous version(s):

Editor(s):

Mark Cresitello-Dittmar

Authors:

Doug Tody, Francois Bonnarel, Omar Laurino, Mireille Louys, Arnold Rots, Jose Enrique Ruiz, Jesus Salgado, and the IVOA Data Model Working Group.

Abstract

Status of This Document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

Acknowledgements

This document has been developed with support from NSF and NASA under the Virtual Astronomical Observatory (VAO) project, the National Science Foundation's http://www.nsf.gov Information Technology Research Program under Cooperative Agreement AST0122449 with The Johns Hopkins University, from the UK Particle Physics and Astronomy Research Council (PPARC) http://www.pparc.ac.uk, and from the European Commission's Sixth Framework Program http://fp6.cordis.lu/fp6/home.cfm via the Optical Infrared Coordination Network (OPTICON), http://www.astro-opticon.org.

Change Log:

2013 May 05: Initial release of ImageDM draft document.
2013 Aug 16: Rev 16 updates.
2013 Nov 15: Second draft of ImageDM release.
Updated to reflect initial reviewer comments. Architecture section extensively revised to provide compatibility with ObsCore/Observation and Char2
2014 Sep 10: Model restructured as extension of Observation Dataset
Extensive revision, basing Cube model as extension of dependent models: Dataset, and STC
Define new components defining generic Cube datasets (image and sparse) which can be utilized for Spectral and other models
Baseline to STC-2.0 prototype model.
2015 Mar 20: Updated STC-2.0 prototype model spec.
2016 Nov 09: Model revisions from implementation
- Generalized to NOT be directly dependent on STC, which should be considered to implement elements in this model (eg: CoordSys)
- relation changes due to type change on coordinate elements.
2017 Feb 03: Model revisions from implementation
- Again modifying STC relation, showing as aggregated relations which will allow the appropriate relations in this model, while not confining STC usage.
- Updated representation of STC sub-models (coordsys, coords, trans)
- A bit of restructuring to pull Dataset-DataProduct relation to forefront, should improve readability.

Contents

1 Introduction

1.1 Motivation

The concept of the *astronomical image* goes back decades, most notably to the introduction of the flexible image transport system (FITS) in the late 1970s by Wells and Greisen. Later papers by Greisen, Calabretta, and others added the capability to define a *world coordinate system* (WCS) that can be associated with an image to define the physical coordinates in an M-dimensional space of each data sample (observable) in the N-dimensional array comprising the data segment of the image. Other metadata elements (FITS keywords) are defined to describe the origins and content of the particular image dataset. The astronomical multi-dimensional image concept is a type of data hypercube, and is related to the volumetric cubes used in medical and geological applications, and to commercial technologies such as the OLAP cube, which projects a RDBMS relation onto the axes of a hypercube.

An important aspect of the astronomical image is *abstraction*. The image model hides how the data are physically stored; this is especially important for large images or image cubes, which may be Gigabytes or Terabytes in size for a single dataset. While logically the data portion of the image may be a simple N-d array, physically the data may be represented or stored in many different ways. Large cubes may be physically stored in multiple smaller segments, or data may be stored in N-d blocks to provide uniform access along any dimension or axis of the image. Sparse cubes may be stored as multiple segments, each at a given location within the larger logical cube. Data may be stored in a compressed form, or may be encoded, e.g., via a multiresolution technique such as a wavelet transform (JPEG2000). Each such representation offers certain advantages and disadvantages; by separating the logical view of the data from the details of how it is physically represented, the optimum choice may be made for each application, transparently to higher-level analysis software.

This model presents an abstracted representation of N-Dimensional cube datasets and serves as a framework on which to construct models for more specialized Astronomical datasets.

1.2 Use Cases

A use case study of multidimensional/cube data performed by the US VAO and community partners in early 2013 contributed to early development of this data model. A special session on multidimensional data held by the IVOA in May 2013 helped to further refine community use cases for the data model.

A comprehensive analysis of use cases for the Image data model in the context of multidimensional astronomical data is beyond the scope of this document, but is available in [TBD: *reference to VAO use case analysis*].

Here we summarize major use cases from this study:

- Simple 2-D image, the most common astronomical image type.
- A single cube image is often sparse in either the spectral or spatial or temporal plane. The JCMT cubes are a good example of a spectral data cube that is sparse in the spatial plane.
- Multiple-subarray images are common for instrumental observations composed of multiple detectors. For example, a spectral data cube with multiple spectral bands, or a

CCD mosaic where the detectors may not be exactly aligned, requiring different WCS calibrations.

- Very large cubes are increasingly common with modern instruments. It becomes physically impossible to store these as a single cube dataset (file); large cubes must be stored as multiple files, often with sophisticated voxel encoding, compression, tiling, or other data representation algorithms. The logical image abstraction needs to hide this complexity from client applications to simplify and unify data access.
- A key use case for using image/cube data is interactive image visualization and analysis, particularly for very large cubes. In this case, the image dataset is initially rendered in 2-D, greatly reduced in size, then the user interactively slices and dices the cube, computes 2-D projections, extracts spectra, computes moments, etc., to interactively view the contents of the cube. For very large cubes, this must be done via remote access to the cube data, which is staged to a parallel cluster providing the parallel computing capability required to quickly render the large cube, that may be tens to hundreds of GB in size or larger.

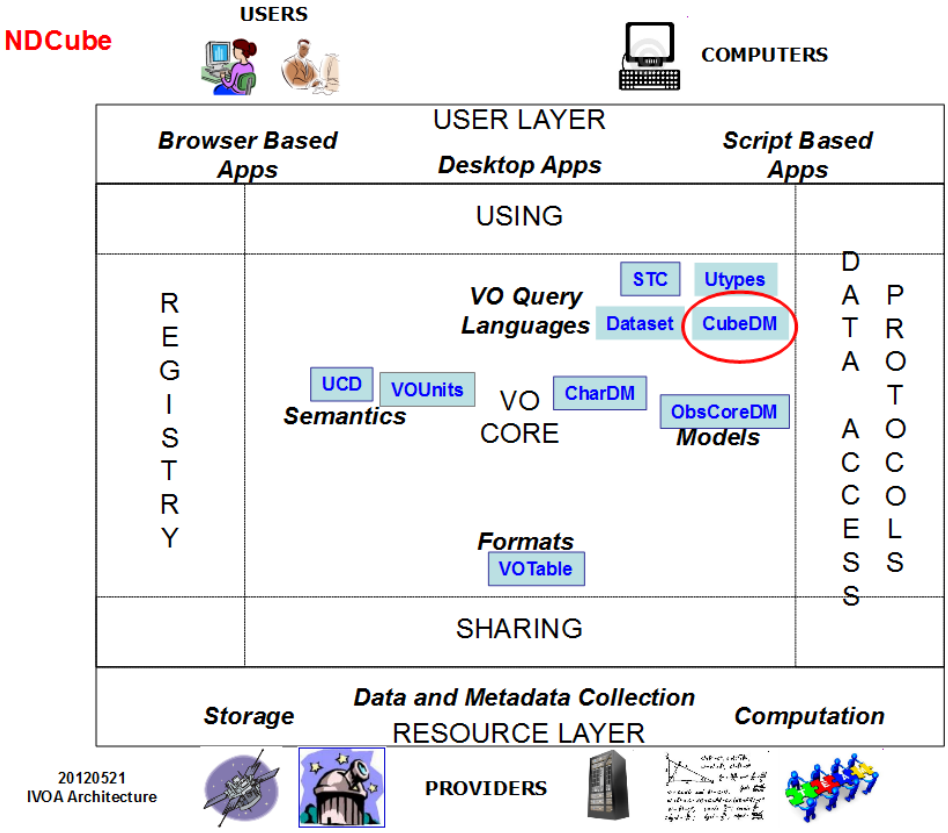
1.3 Cube Classifications

From the primary use cases, we can define these classifications of cube data to be addressed by this model:

1. **Simple image**. A single filled or mostly filled N-Dimensional image array with associated metadata describing the dataset, its WCS coordinate systems, etc.
2. **Simple sparse image.** A single N-Dimensional array as in case #1, however, large portions of the image may be sparse (have no data samples).
3. **Multiple subarray image.** An image dataset containing multiple subarrays, i.e., N-Dimensional image data arrays within the coverage of the overall image dataset. The subarrays may differ in size, resolution, coverage, or other characteristics. The overall image dataset containing the subarrays may be sparse. The overall image dataset does not have an explicit image geometry or sampling, only coverage. There are two sub-cases here:
   a) the subarrays are all part of the same observation and share common metadata. For example, a multi-band image observation.
   b) the subarrays may differ arbitrarily and the overall image dataset is essentially an aggregation of (possibly sparse) simple images as in case #1 or #2. For example, a complex image aggregating data from multiple observations.
4. **Large cube.** Very large cubes may be represented as either a simple image or multi-subarray type. The cube model abstraction does not impose any size limitations. The important factor here, is that the chosen view of the cube is independent of the physical storage structure.
5. **Wide-field survey.** This is essentially the same as #3, except that no subarrays (no explicitly pixelated data) or individual image datasets may be externally visible. The survey has coverage, but only automated virtual data generation techniques may be used to access the data, with sub-regions being computed on-the-fly and returned to the client. Alternatively, a survey might expose a collection of discrete cubes, in which case #1 or #3 may be used.
6. **Sparse data cube.** Sparse data are commonly used for higher-dimensional cubes, and are frequently sparse along one or more axes. For example, a multi-band image has

data at only a few given spectral coordinates, (each corresponding to a spectral bandpass). A spectral (or velocity) data cube may contain data for a number of widely spaced spectral bands, each of which may differ in the spectral resolution and number of channels. A time cube likewise may contain data, either individual points, or time series, arbitrarily spaced along the time axis with time regions where no data was taken. A multi-object spectral data cube may be sparse in the spatial plane. Event data can be considered a data cube which is sparse in all measurement axes.

1.4 IVOA Architecture Context



1.5 Model Dependencies

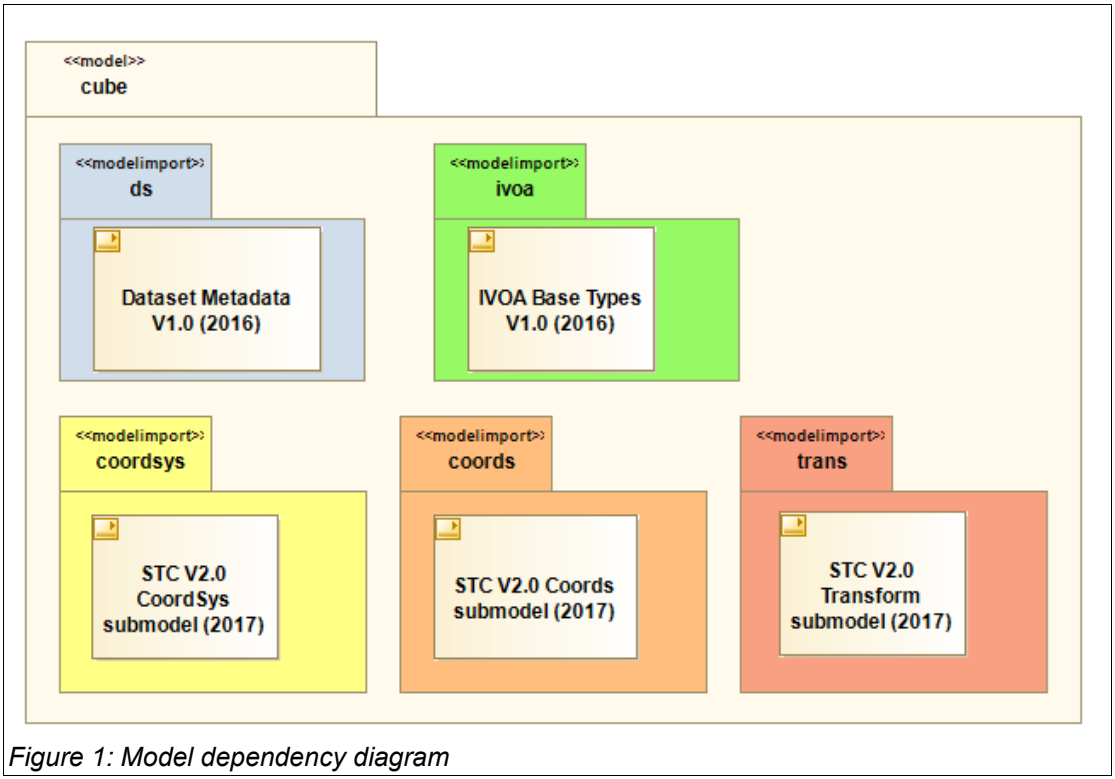


Figure 1: Model dependency diagram

The ND-Cube model is built on other IVOA data models as indicated in Figure 1. The <<model>> and <<modelimport>> stereotypes provide information identifying the model, it's version, any dependencies, and URLs to find more information about the model definitions including HTML and schema documentation. See Appendix B for more information about the content of these stereotypes and how they are used in serializations.

1.6 Structure of this Documentation

+ Major sections for each model area (Dataset, Observation, etc.).
+ First subsection in each section is the primary element within that model
+ Subsequent subsections for secondary elements, in alphabetical order.
+ Each subsection has sub-subsections for each attribute/relation
    - attributes show the full definition including datatype and usage.
    - relations describe the usage of the object in that context, the type of the target of the relation, and a reference to the full definition of that type.

2 DataProduct

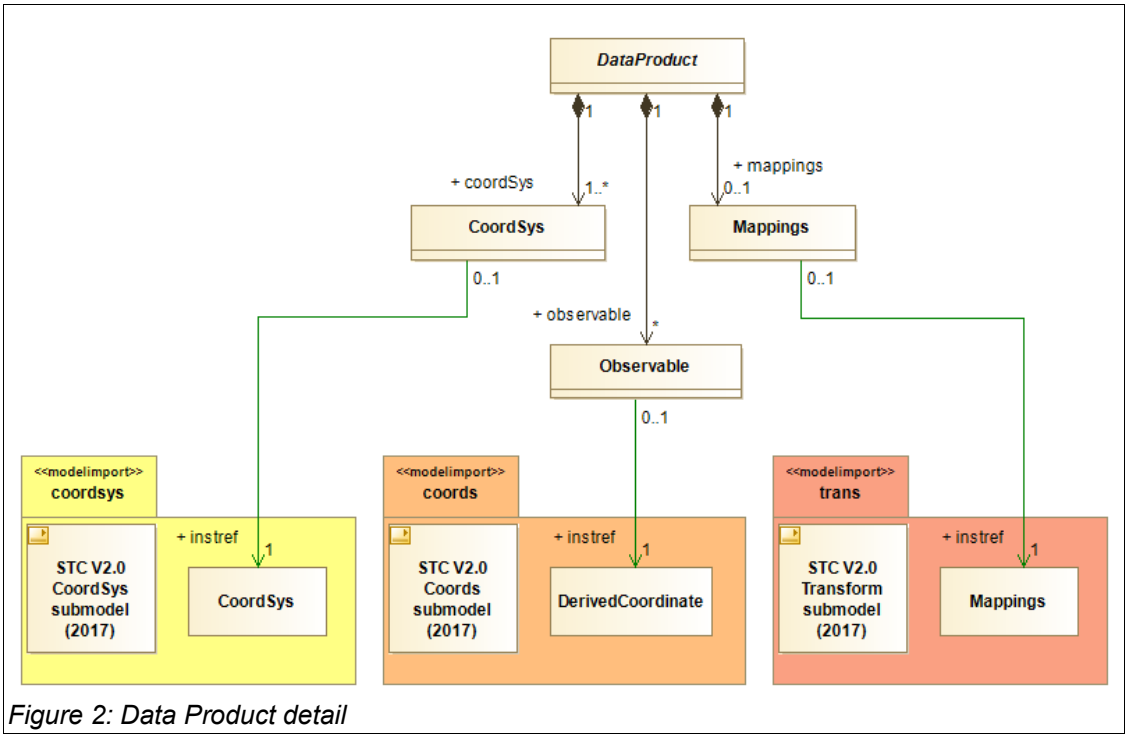


Figure 2: Data Product detail

Data Products represent a single instance of a particular type of data. It contains the minimum set of metadata required to define the data product type, all other metadata is left to the Dataset. The Data product definition is constant, regardless of what entity or process is generating it. For example, a Spectrum data product has the same structure and content regardless of whether it is a result of an observation, simulation, or fit to a theoretical model.

2.1 DataProduct

Abstract head of the data product tree. All astronomical data products are expected to define the set of coordinate systems associated with the data, and any mappings which define the relations between coordinate frames. Typically mappings are used to define transformations between a pixelated coordinate frame and their corresponding world coordinate frame, but this is not always the case. For example, in sparse cube data such as event lists, mappings may be defined between physical frames (chip or sky) to world coordinate frames (csc or eqpos).

2.1.1 DataProduct.coordSys

type: CoordSys **type-detail: Section 2.2**
multiplicity: 1..*

One or more coordinate systems associated with a particular data product.

2.1.2 DataProduct.mappings

type: Mappings **type-detail: Section 2.3**
multiplicity: 0..1

The Mappings object holds a collection of frame transform definitions which facilitate the conversion of coordinates and other objects from one frame representation to another. For example, convert pixel coordinates from the pixel space to world coordinates in a 2D space frame.

2.1.3 DataProduct.observable

type: Observable **type-detail: Section 2.4**
multiplicity: 0..*

The DataProduct owns each data sample (observable) it contains. For particular DataProducts, objects may be defined which refer to these elements and provide structure and context to the individual sample instances. For example, a Point containing a collection of observables taken at the same time. NOTE: This object primarily defines ownership and relation of the data to the data product, however, in practice, access to these objects would typically be through these organizational objects which provide the necessary structure.

2.2 CoordSys

The CoordSys object is a container for coordinate system definitions. This element uses the vo-dml 'aggregation pattern' to indicate that the DataProduct owns its instances of this object, but allowing for other objects do so as well.

2.2.1 CoordSys.instref

type: coordsys:CoordSys **type-detail: n/a**
multiplicity: 1

This indicates that the instances of CoordSys contained by the DataProduct are defined according to the coordsys model element 'CoordSys'.

2.3 Mappings

The Mappings object is a container to manage frame transform definitions which are then available for use by other objects. This element uses the vo-dml 'aggregation pattern' to indicate that the DataProduct owns its instances of this object, but allowing for other objects to do so as well.

2.3.1 Mappings.instref

type: trans:CoordSys **type-detail: n/a**
multiplicity: 1

This indicates that the instances of Mappings contained by the DataProduct are defined according to the transform model element 'Mappings'.

2.4 Observable

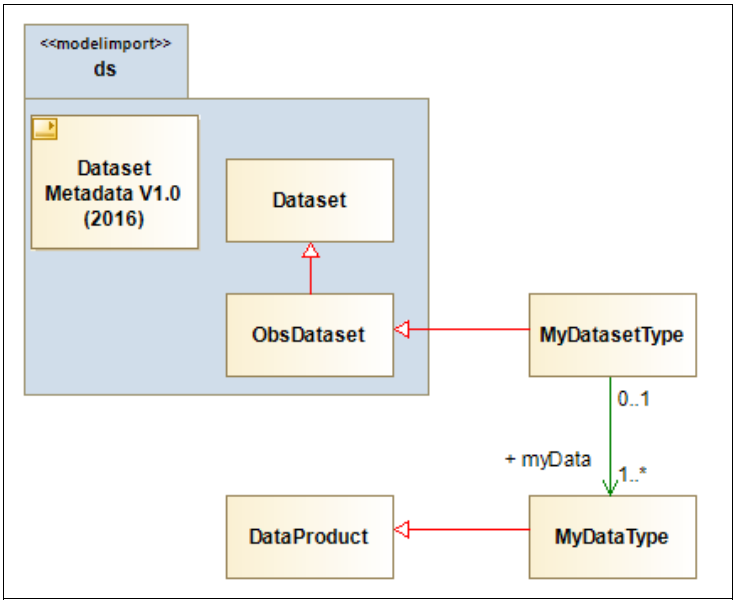
The Observable object is a container for coordinate definitions. This element uses the vo-dml 'aggregation pattern' to indicate that the DataProduct owns its instances of this object, but allowing for other objects to do so as well.

2.4.1 Observable.instref

type: coords:DerivedCoordinate **type-detail: n/a**
multiplicity: 1

This indicates that the instances of Observable contained by the DataProduct are defined according to the coordinates model element 'DerivedCoordinate'.

3 Dataset-DataProduct Composite Patern



To define a complete Dataset/DataProduct requires both components. The Dataset providing descriptive, identification, and provenance metadata related to how the dataset was generated, and one or more DataProduct-s with the data content. This pattern facilitates the description of particular data products generated by different means, as well as complex datasets composed of multiple data products (by defining a DataProduct which is itself a Dataset).

In the next section, we use this pattern to define NDCube datasets.

4 N-Dimensional Datasets

4.1 NDImageDataset

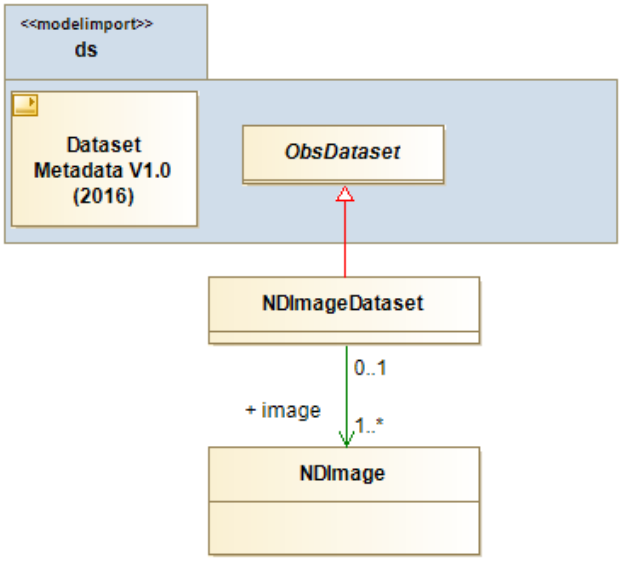


Figure 3: N-Dim. image dataset detail

The NDImageDataset is an extension of the Observation-Dataset (ObsDataset) object of the Dataset Metadata model which provides all the generic IVOA Dataset metadata (DataID, Curation, etc.), plus metadata related to observational data (Target, Characterisation, ObsConfig, etc.). See sections 2 and 3 of the Dataset Metadata document for a full description of the inherited content.

The IVOA Dataset is defined as one or more files which are considered to be a single deliverable. The NDImageDataset, should be considered a collection of one or more NDImage objects, each of which defines a particular image instance.

In terms of our image cube classifications:

- **Simple Image** - a single NDImageDataset with identifying metadata, and a single NDImage instance containing the data array definitions, coordinate systems, and (WCS) mapping definitions.
- **Single Sparse Image** - is the same as Simple Image, except in the mapping definitions used to transform the pixel space to world coordinates.
- **Multiple Subarray Image** - Use case 3a consists of a single NDImageDataset with multiple subarrays, each of which is a separate NDImage instance. The dataset instance contains metadata for the entire image dataset. Each NDImage instance contains an N-Dimensional image array with associated metadata, including coordinate mappings, related to that instance. Use case 3b is an aggregation of related but otherwise independent Image instances. This is a form of *complex dataset* wherein standard

datasets are combined to model more complex data. How the individual datasets are related is defined by the application and is not within the scope of this model.

4.1.1 NDImageDataset.image

type: NDImage **type-detail: Section 6.1**
multiplicity: 1..*

Reference to one or more NDImage instances. Each instance represents an individual image array, complete with metadata describing its pixel space, coordinate systems, and the mappings between coordinate frames.

4.2 Sparse Cube Dataset

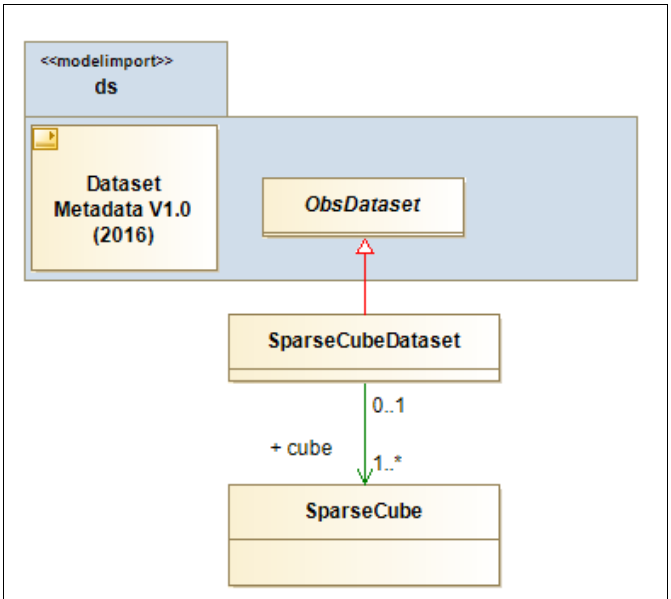


Figure 4: Sparse Cube Dataset detail

The SparseCubeDataset is an extension of the Observation-Dataset (ObsDataset) object of the Dataset Metadata model which provides all the generic IVOA Dataset metadata (DataID, Curation, etc.), plus metadata related to observational data (Target, Characterisation, ObsConfig, etc.). See sections 2 and 3 of the Dataset Metadata document for a full description of the inherited content.

The SparseCubeDataset is suitable for encapsulating N-Dimensional data which are sparse in nature, and do not have a set of regularly sampled (pixel) axes. For example, an event list.

4.2.1 SparseCubeDataset.cube

type: SparseCube **type-detail: Section 7.1**
multiplicity: 1..*

Reference to one or more SparseCube instances. Each instance represents an individual cube, complete with metadata describing its coordinate systems, and the mappings between coordinate frames.

5 N-Dimensional Data Products

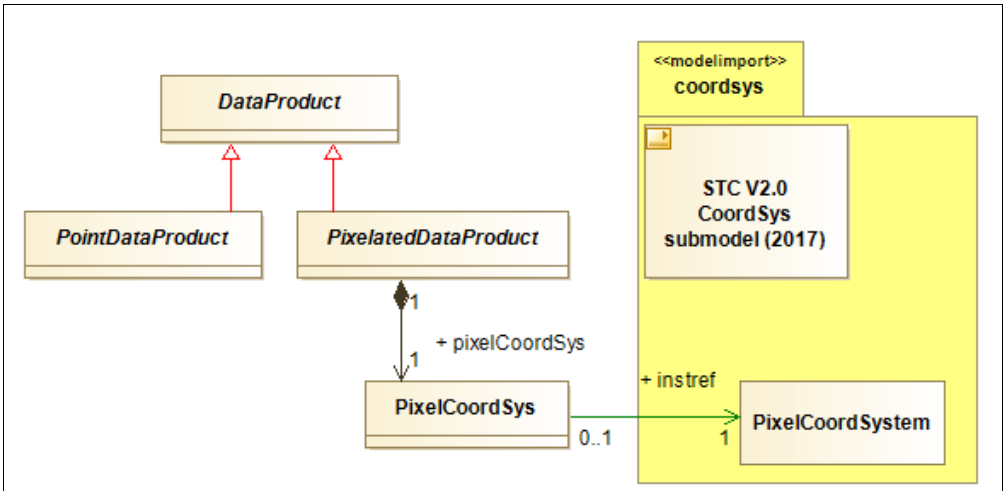


Figure 5: NDimensional Data Product detail

We divide the description of N-Dimensional data products into two primary categories, one for point-like data, and another for pixelated data, which adds the specification of the pixel coordinate system.

5.1 PixelatedDataProduct

Abstract extension of DataProduct specialized for pixelated (image) data. In addition to world coordinate systems, a pixelated data product must also define the pixel coordinate system and the pixel space which the image occupies.

5.1.1 PixelatedDataProduct.pixelCoordSys

type: PixelCoordSys **type-detail: Section 5.3**
multiplicity: 1

The PixelCoordsys object holds the description of the pixel frames, and axis groupings. It also contains the PixelSpace object, which defines the number and span of each pixel axis.

5.2 PointDataProduct

Abstract extension of DataProduct specialized for point data.

5.3 PixelCoordSys

The PixelCoordSys object provides the pixel domain specification. Typically, this will include the pixel frame definitions for the image, and a description of the pixel space (the number and span of each pixel axis). We model this element according to the vo-dml 'aggregation pattern'. This indicates that the PixelatedDataProduct owns its instances of this element, but allows other objects to do so as well.

6 N-Dimensional Image

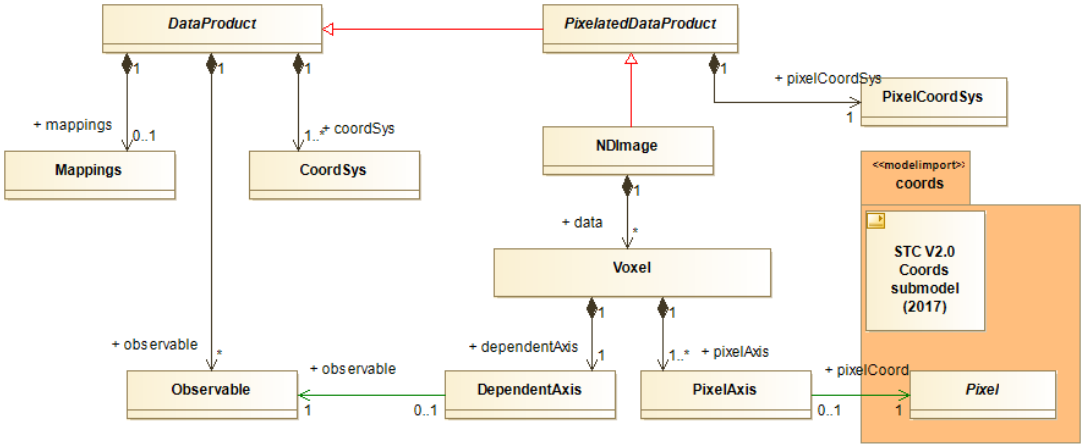


Figure 6: NDImage detail

6.1 NDImage

The essential characteristic of an "image" dataset is the presence of a multidimensional, regularly sampled numerical array, with associated metadata describing the image instance. While the concept of an image as a multidimensional array is fully general, astronomical datasets have some combination of specific "domain" related axes (spatial, spectral, time, polarization, etc), with flux or some other derived value, as the array element (voxel) value. The mapping of image pixel axes to physical coordinate frames is described by a set of transformation definitions associated with the image.

Unless dimensionality is otherwise indicated, the terms *image*, *cube*, and *hypercube* are interchangeable and refer to image (array-valued) data of arbitrary dimension. Image is a specialized case of general *hypercube* or *n-cube* data where the value at a given point in the hypercube is restricted to a simple numerical value. The data samples of an image are referred to as *pixels* (picture elements) or as *voxels* (volume elements), pixels being the preferred term for 2D images.

6.1.1 NDImage.data

type: Voxel **type-detail: Section 6.2**
multiplicity: 0..*

The NDImage contains a set of zero or more voxels, holding the pixel coordinates in N-D space, and the associated value at that location.

6.2 Voxel

The 'voxel' or volume element, is a fundamental atom on an N-Dimensional image. It is defined by a pixel coordinate, locating the volume in pixel space, and the value at that location.

6.2.1 Voxel.pixelAxis

type: PixelAxis **type-detail: Section 6.4**
multiplicity: 1..*

One or more pixel axis which locates the voxel in N-D pixel space.

6.2.2 Voxel.dependentAxis

type: DependentAxis **type-detail: Section 6.3**
multiplicity: 1

Identifies the Observable data sample associated with the voxel. ie: the 'value' of the Voxel.

6.3 DependentAxis

The DependentAxis identifies the Observable data sample, or 'value' associated with a Voxel.

6.3.1 DependentAxis.observable

type: Observable **type-detail: Section 2.4**
multiplicity: 1

Reference to the Observable data sample value for the Voxel.

6.4 PixelAxis

The PixelAxis identifies a pixel coordinate location associated with the voxel.

6.4.1 PixelAxis.pixelCoord

type: coords:Pixel **type-detail: n/a**
multiplicity: 1

Reference to the pixel coordinate value for the Voxel along a particular pixel axis or axis set.

7 Sparse Cube

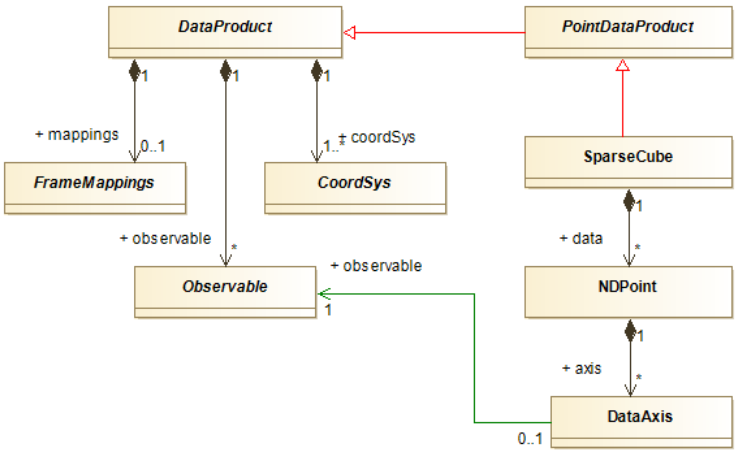


Figure 7: Sparse Cube detail

7.1 SparseCube

An extension of PointData, the SparseCube is a collection of N-Dimensional points, each of which provide a value for the point in each represented domain. Any domains which are not represented by the specialized domain types, can provide values through the customAxes element.

7.1.1 SparseCube.data

type: NDPoint **type-detail: Section 7.2**
multiplicity: 0..*

Zero or more NDPoints, providing the sparsely sampled data values in each represented domain.

7.2 NDPoint

The NDPoint object is defined by collection of values on each axis. As a unit, these completely define a Point in N-Dimensional coordinate space.

7.2.1 NDPoint.axis

type: DataAxis **type-detail: Section 7.3**
multiplicity: 0..*

Zero or more DataAxis objects, each identifying the observable data sample for a particular axis of the Point.

7.3 DataAxis

The DataAxis object provides a sorting mechanism for associating observable values of a particular domain or axis.

7.3.1 DataAxis.observable

type: Observable **type-detail: Section 2.4**
multiplicity: 1

Reference to the observable data sample for a particular data axis.

8 Data Types

8.1 IVOA Data Types

The ivoa model provides a set of standardized primitive data types as well as types for representing quantities (values with associated units and ucd). We provide a diagram of the model here, and refer the reader to Appendix F of the VO-DML modeling specification document[8] for more information.

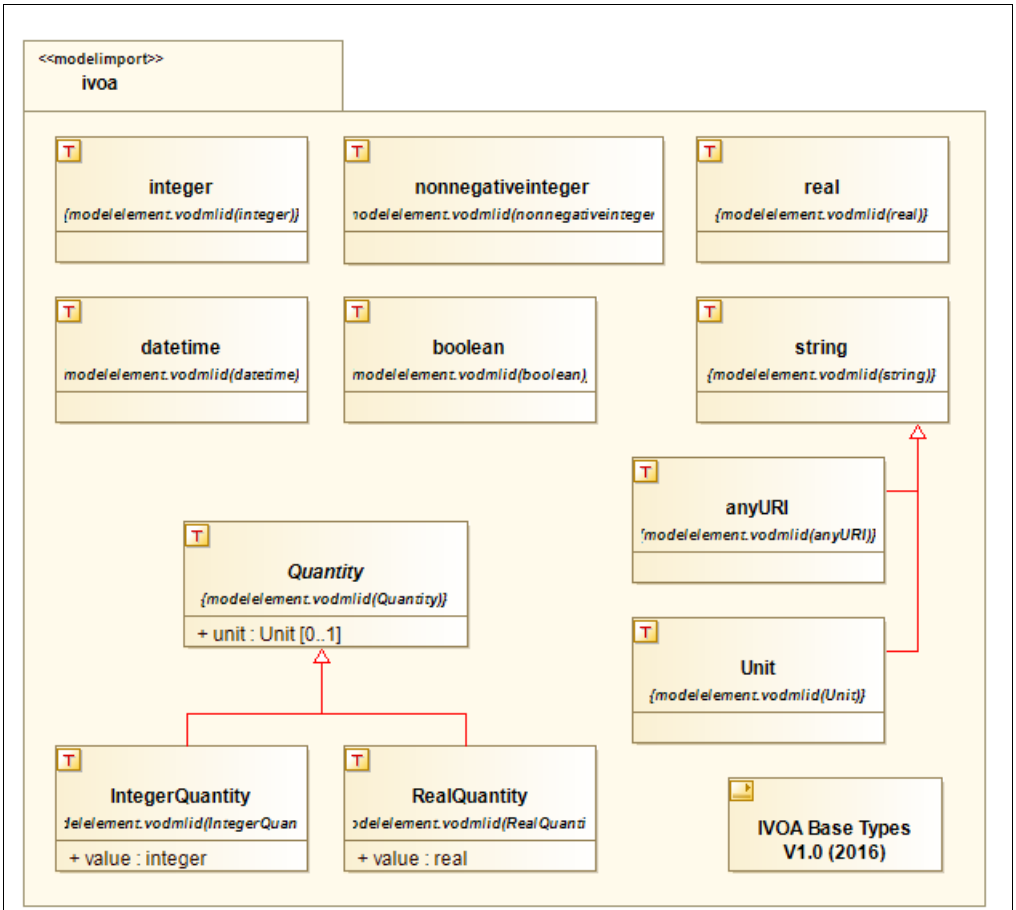


Figure 8: IVOA Base Data Types

8.1.1 Units

This model requires the use of the IVOA VOUnits Standard[9] for representing units of physical quantities. This standard reconciles common practices and current standards for use within the IVOA community.

8.1.2 UCDs

This model requires the ucd field to comply with syntax defined in ”An IVOA Standard for Unified Content Descriptors”[11].

8.1.3 Dates

The 'datetime' datatype is for expressing date-time values. The string representation of a datetime value should follow the FITS convention for representing dates. The FITS standard is effectively ISO8601 format without the "Z" tag to indicate UTC (YYYY-MM-DDThh:mm:ss). Values are nominally expressed in UTC.

8.2 Cube Model DataTypes

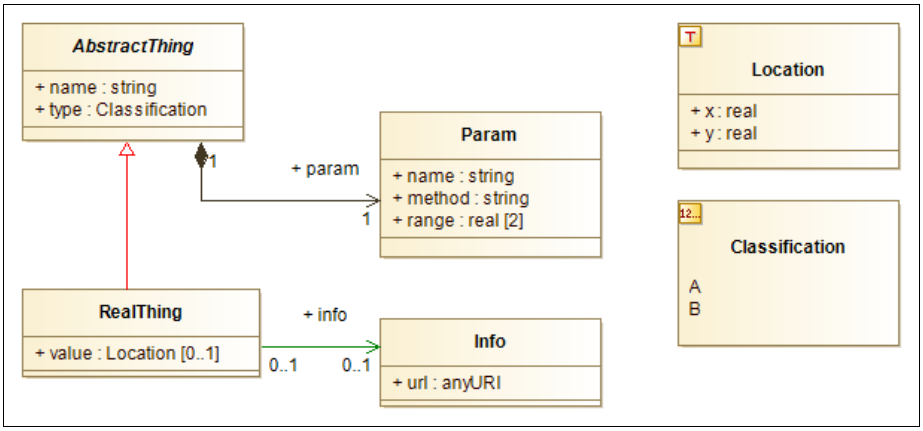
This model defines no new data types or enumerations.

Appendix A: Modeling Conventions

This model adopts the same modeling conventions as outlined in Appendix A of the Dataset Metadata model. Here, we provide a brief summary of these statements.

1 Diagram notation

This model follows the VO-DML modeling practices, however, UML representations may vary depending on the tool used. Below, we describe the graphical representation of the modeling concepts and relations.



1.1 Class

Classes are represented by a plain box. The class name is annotated in the top window, abstract classes use italic typeface. Attributes, if any, are listed in the lower panel. Attributes may only be of primitive type (real, string, etc), a defined DataType, or an Enumeration type. Relationships to other objects are defined via the composition and reference relation arrows.

1.2 DataType

DataTypes are represented by a box shape similar to Class, but annotated with a "T" symbol in the top left corner.

1.3 Enumerations

Enumerations are represented by a box shape similar to Class, but annotated with a "1,2.." symbol in the top left corner. Enumeration Literals (possible values) are listed below the enumeration class name.

1.4 Generalization

Generalizations are represented by a red line, with open triangle at the end of the source, or more general, object.

1.5 Composition

The composition relation is indicated by a black line with a solid diamond attached to the containing object, and an arrow pointing to the object being contained. The composition relation is very tight, where the container is responsible for the creation and existence of the target. Any object may be in no more than one composition relation with any container. The attribute name for the composition relation is annotated at the destination of the relation (e.g. "+ dataID"). This is typically a lower-cased version of the destination class name, but this is not required.

1.6 Reference

The reference relation is indicated by a green line, with an arrow pointing to the object being referenced. The reference relation is much looser than composition, the container has no ownership of the target, but merely holds a pointer, or other indirect connection to it. The attribute name is annotated at the destination of the relation (e.g. "+ proposal"). This is typically a lower-cased version of the destination class name, but may be another name indicating the role that the class is playing in this context.

1.7 Multiplicity

All attributes and relations have a multiplicity associated with them. For attributes, the multiplicity is contained within brackets just after the attribute name. If no bracket is displayed, this is equivalent to '[1]'.

+ 1 = one and only one value must be provided.
+ 0..1 = zero or one value may be provided.
+ * = zero or more values may be provided (open ended).

2 Model Identification metadata

We defined stereotypes on elements of this model to provide metadata identifying this model, and other models on which it is dependent.

2.1 Model stereotype

The Model stereotype (<<Model>>) consists of a set of Model properties which identify a particular model and its dependencies.

2.2 Import Stereotype

The <<import>> stereotype is attached to Packages representing imported models. It identifies the model by name, and provides URLs from which the full description may be obtained.

3 Extensibility

User-defined content would be modeled as an extension of the IVOA standard.

3.1 Scope

We permit any object modeled in this document to be extended with user-defined content, with the following restrictions:

- Follow VO-DML modeling practices.
- Values of extended content must be consistent with the content of modeled data. That is, using the IVOA base primitive types, Quantity, and STC Coordinates as appropriate.
- Since extended content, by definition, does not follow the corresponding model, it is not possible for general applications to interpret complex structures within that content. It is, therefore, recommended that users define extended content in such a way as to avoid ambiguity between its components.

3.2 Support

Applications should, but are not required to, provide the following support for extended content:

- Retain existence of extended content, including namespace and UTypes.
- Retain association with modeled component.
- Provide access to extended content by users.

Appendix B: N-Dimensional Cube Model Summary

Cube Model Identification				
Model Element	**Datatype**	**Mult.**	**Meaning**	**value**
Model identification				
Model				
Model.name	string	1	Data model name	"N-Dimensional cube"
Model.version	string	1	Data model version	"1.0"
Model.prefix	string	1	Data model prefix tag	"cube"
Model.url	anyURI	1	Reference URL for model	<TBD>
Imported Model				
Import.name	string	1	Imported model name	"ds"
Import.version	string	1	Imported model version	"1.0"
Import.url	anyURI	1	Reference URL for imported model	<TBD>
Imported Model				
Import.name	string	1	Imported model name	"coordsys"
Import.version	string	1	Imported model version	"2.0"
Import.url	anyURI	1	Reference URL for imported model	<TBD>
Imported Model				
Import.name	string	1	Imported model name	"coords"
Import.version	string	1	Imported model version	"2.0"
Import.url	anyURI	1	Reference URL for imported model	<TBD>
Imported Model				
Import.name	string	1	Imported model name	"trans"
Import.version	string	1	Imported model version	"2.0"
Import.url	anyURI	1	Reference URL for imported model	<TBD>
Imported Model				
Import.name	string	1	Imported model name	"ivoa"
Import.version	string	1	Imported model version	"1.0"
Import.url	anyURI	1	Reference URL for imported model	<TBD>

Cube Model Summary				
Model Element	**Datatype**	**Mult.**	**Meaning**	**UCD1+**
Cube Model Elements				
CoordSys			STC2 CoordSys model CoordSys	
DataProduct			Instance of data product (abstract)	
DataProduct.coordSys	CoordSys	*	Coordinate systems associated with data	
DataProduct.mappings	Mappings	0..1	Coordinate frame transforms	
Mappings			STC2 Transform model Mappings	
NDImageDataset			Instance of N-Dimensional image cube dataset	
NDImageDataset			Instance of N-Dimensional image dataset	
NDImageDataset.image	NDImage	1..*	N-Dimensional image arrays	
NDImage			N-Dimensional Image data	
NDImage.data	Voxel	*	N-Dimensional Image data array	
NDPoint			N-Dimensional data point	
NDPoint.axis	DataAxis	0..1	Point dimension axis	
PixelCoordSys			STC2 CoodSys model Pixel coordinate system	
PixelatedDataProduct			Instance of pixelated image data product (abstract)	
PixelatedData.pixelCoordSys	PixelCoordSys	1	Pixel coordinate system defintion.	
PointDataProduct			Instance of sparse data product (abstract)	
SparseCubeDataset				
SparseCubeDataset.cube	SparseCube	1..*	Instances of sparse cube data	
SparseCube			Instance of sparse cube data	
SparseCube.data	NDPoint	*	N-Dimensional sparse cube points	
Voxel			Volume pixel	
Voxel.pixelAxis	PixelAxis	1..*	Voxel pixel dimension axis	
Voxe.dependentAxis	DependentAxis	1	Voxel Observable axis dimension	

References

[1] "IVOA Dataset Metadata Model": Version 1.0, 10 Sept. 2014
http://www.ivoa.net/Documents/...
[9] ”Units in the VO”: Version 1.0
http://www.ivoa.net/Documents/VOUnits/20120820/PR-VOUnits-1.0-20120820.pdf
[10] ”The UCD1+ controlled vocabulary”: Version 1.23
http://www.ivoa.net/Documents/REC/UCD/UCDlist-20070402.pdf
http://cdsweb.u-strasbg.fr/UCD/ucd1p-words.txt
[11] ”An IVOA Standard for Unified Content Descriptors”: Version 1.10
http://www.ivoa.net/Documents/REC/UCD/UCD-20050812.pdf
http://www.ivoa.net/Documents/latest/UCD.html
[8] ”VO-DML a consistent modeling language for IVOA data models”: Version 1.00-20140427
http://volute.googlecode.com/svn/trunk/projects/dm/vo-dml/doc/VO-DML-WD-v1.0.pdf