*I*nternational

*V*irtual

*O*bservatory

*A*lliance

IVOA Image Data Model

Version 1.0

IVOA Working Draft 2013-08-12

This version:
WD-ImageDM-1.0-20130811
Latest version:
http://www.ivoa.net/Documents/ImageDM
Previous version(s):

Editor(s):
Douglas Tody
Francois Bonnarel

Author(s):
Douglas Tody
Francois Bonnarel
Mark Cresitello-Dittmar
Mireille Louys
Arnold Rots
Jose Enrique Ruiz
Jesus Salgado

Abstract

The Image Data Model (**ImageDM**) describes datasets characterized by an N-dimensional, regularly sampled numeric data array with associated metadata describing an overall multi-dimensional "*image*" dataset. Image datasets with dimension greater than 2 are often referred to as "*cube*" or "*image cube*" datasets and may be considered examples of *hypercube* or *n-cube* data. In this document the term "image" refers to general multi-dimensional datasets and is synonymous with these other terms.

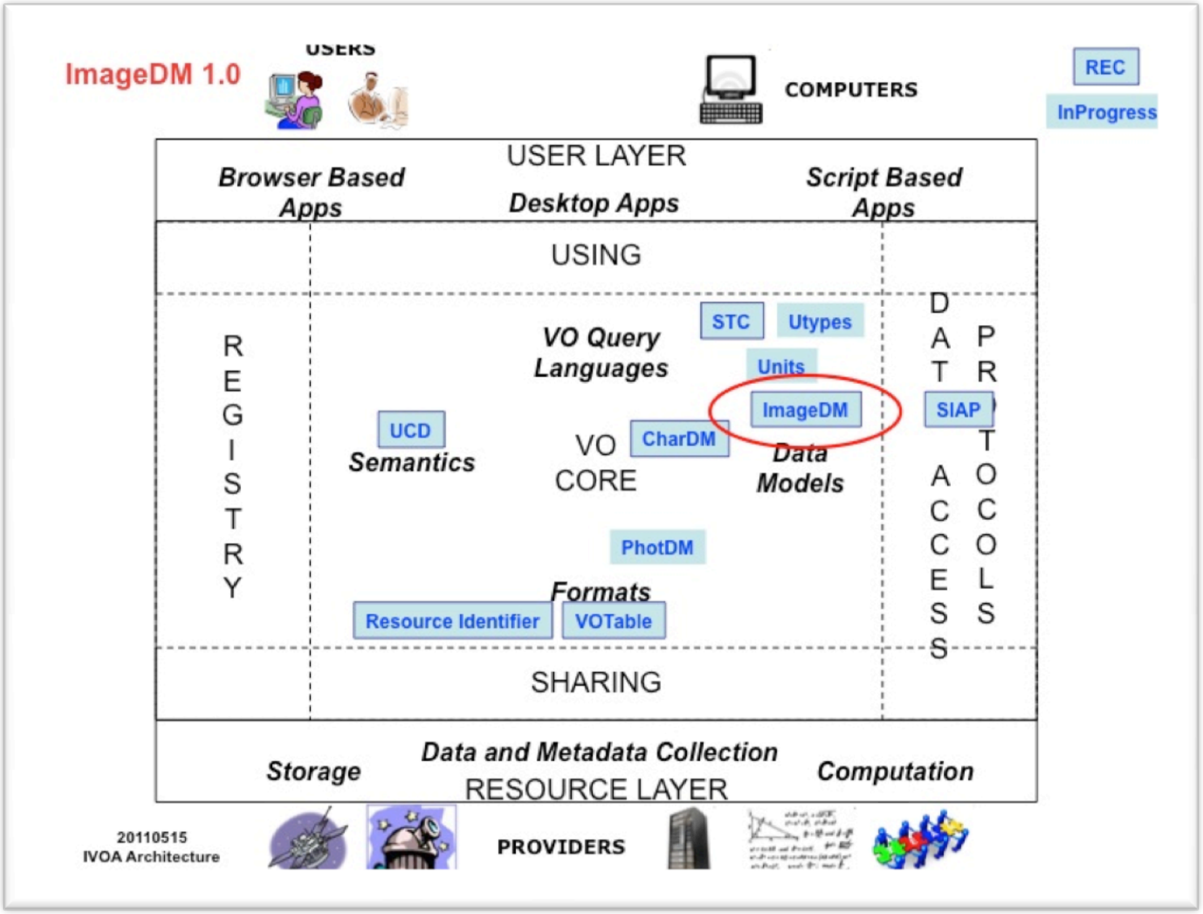
An image may have an associated *world coordinate system* (WCS) associating physical coordinates with each measurement axis. The standard axis types for astronomical data are the physical attributes of an observable, i.e., spatial (including celestial projections), spectral (including redshift and velocity), time, and polarization. The ImageDM encompasses all such multidimensional, regularly sampled, array-valued data where the WCS describing the attributes of each data sample is separable from the data array. Support for sparse data is included. Although regularly sampled, N-dimensional array-valued data are emphasized, limited support for more fundamental multi-dimensional observational data including event and visibility data is included.

The ImageDM is related to other IVOA Data models (Observation DM, ObsCore DM, Characterization DM, and Spectral DM), and is intended to serve as the basis for VO data access protocols such as SIA (Simple Image Access) and TAP/ObsTAP (table access, indexing of observational data products) when the latter is used to access image data. Support is included for both 2-D images and multidimensional cubes, as well as very large (Terabyte-sized) cube datasets.

As with most of the VO Data Models, ImageDM makes use of other VO data models including FITS, STC, Utypes, Units and UCDs. ImageDM instances are serializable in a variety of data formats including but not limited to FITS (for n-D image dataset instances) and VOTable (primarily for discovery queries). Additional data formats such as HDF5 and JPEG2000, and environment-specific formats such as CASA image tables and Starlink NDF are also considered, and can provide better performance for large cube datasets.

Link to IVOA Architecture

The figure below shows where the Image DM fits within the IVOA architecture:



The ImageDM depends upon the Characterization data model (CharDM), the Photometry data model (for specifying spectral bands), as well as the Utype mechanism and standard, the Space-Time coordinate metadata (STC), and the UCD mechanism. The ImageDM is also based upon the FITS image model with associated FITS WCS.

Status of this Document

The first release of this document was 2013 May 05.

This document has been produced by the authors and the IVOA Data Model Working Group. The current document is a working draft intended only to advance the design of the IVOA Image data model and support related prototyping. It is not yet sufficiently developed and stable to support external review or usage.

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

Acknowledgements

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.

Contents

1 Introduction

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.

A key aspect of the astronomical image is separation of metadata such as the WCS from the data array, which is a simple N-dimensional array of numerical data values. Representing the data portion as a simple N-dimensional numerical array is important for computational efficiency as well as for storage optimization and flexibility. The image model abstraction 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. Representing the data as a simple multidimensional numeric array allows generic software tools to be used for computation, e.g., the array processing capabilities of various scientific languages, or related tools such as *NumPy* in Python. The N-d array (including cube data via various techniques) is relatively easy to render graphically.

Information may be lost in the process of "imaging" an astronomical dataset but the advantages in terms of efficiency and the ability to use generic tools to process and visualize the data often outweigh the loss of information. A multi-level approach can mitigate the problem, using an imaged version of the dataset for initial interaction with the data, with the ability to "drill-down" to the more fundamental data (e.g., event or visibility data) for more precise analysis.

An important aspect of the astronomical image is *abstraction*. 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 multi-resolution 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.

1.1 The Image Data Model and FITS

Given that FITS is so widely used within astronomy for image data, one might reasonably ask why we need a VO Image data model; why not just use FITS instead? FITS actually is used directly within VO; it has been adopted as a core VO technology, used mainly as an efficient binary representation for moderate sized image datasets as well as tables. In the case of image data, FITS is used mainly to represent image datasets returned by a VO service to a client application. In terms of data models, FITS defines a general multi-dimensional image data model and associated WCS model, both of which have been widely adopted within astronomy.

Separation of abstract data model from representation. The main limitation of FITS in a VO context is the lack of separation of the abstract model (e.g., a WCS) from the representation (8 character FITS keywords encoded as 80 character card images)). In a typical VO scenario, a client application or user queries for a list of candidate image datasets matching some client-specified criteria, then selects datasets of interest for retrieval. The standard form for the query response is a VOTable (XML), whereas image datasets are most commonly returned as FITS images. The same image metadata are needed in both cases although the representation is quite different. More generally, the optimal choice of data format depends upon the application, and one may want to serialize image datasets in data formats other than FITS (HDF5, CASA image tables, JPEG, etc.), presenting the same logical data object in each case regardless of the serialization.

Standard VO metadata. VO metadata are richer than what is defined in the FITS standards (although FITS is often extended via (non) standard conventions to model more complex data objects). This is necessary to support uniform data discovery as well as to model specific classes of data such as images, spectra, time series, SEDs, and so forth. The data model for a specific class of data such as the N-d image addressed by the ImageDM inherits from the more generic VO data models such as Observation and Characterisation. The data model for a specific class of data such as Image also needs to be extended to model the unique characteristics of the new class of data. In the case of ImageDM, the WCS submodel is an example of such an extension required for N-d image data. A convention for representing sparse data, particularly important for large higher-dimensional cubes, is another.

Summary. The strategy for developing the VO ImageDM is to capture the most important elements of the FITS image and WCS models, while also providing compatibility and re-use of the relevant VO data models and VO data modeling framework. In particular it should be possible to describe an image dataset in the ImageDM and convert to and from the equivalent FITS image, meanwhile describing the image dataset in a VOTable-based discovery query. Further, it should be possible to serialize an ImageDM instance in other formats and encodings, making it possible to address new use cases such as very large cubes and VO data discovery and virtual data generation, while preserving the semantics of the major FITS models such as the N-d image and associated WCS, leveraging the large investment in FITS by both astronomical software and astronomical data archives.

1.2 Use Cases

A comprehensive analysis of use cases for the Image data model in the context of multi-dimensional astronomical data is beyond the scope of this document, but is available in [reference]. From a design point of view the primary use cases addressed by the ImageDM are the following:

1. **Simple image.** A single filled or mostly-filled n-D image array with associated VO, WCS, and other metadata.
2. **Single sparse image.** A single n-D image 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-D 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: **3a**) the subarrays are all part of the same observation and share common metadata (for example a multi-band image observation), **3b**) 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 by either the simple image or multi-subarray models. The image data model abstraction does not in itself impose any size limitation. The important thing is that the logical image model is separate from how data are actually stored on the back-end.
5. **Wide-field survey.** This is essentially the same as #3 except that no subarrays (no explicitly pixilated 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 subregions 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.

Sparse data are an especially important use case for higher-dimensioned cubes, which are frequently sparse along one or more axes. For example, a multi-band image has data at only a few given spectral coordinates (each actually 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 special case of image data (stretching the concept a bit), which are sparse in all measurement axes.

2 Image Data Model

The essential characteristic of an “image” dataset is the presence of a multi-dimensional, regularly sampled numerical array with associated metadata describing the dataset instance. While the concept of an image dataset as a multi-dimensional array is fully general, astronomical image datasets typically derive from observational data and hence have some combination of spatial, spectral (including velocity or redshift), time, and polarization measurement axes, with flux or some other derived value as the array element value. The mapping of image axes to physical coordinates in measurement space is described by a World Coordinate System (WCS) sub-model, referred to herein as the *Mapping* model. Real world image datasets are not limited to single n-D filled arrays, however the n-D numerical array is at the core of the Image data model (ImageDM).

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.

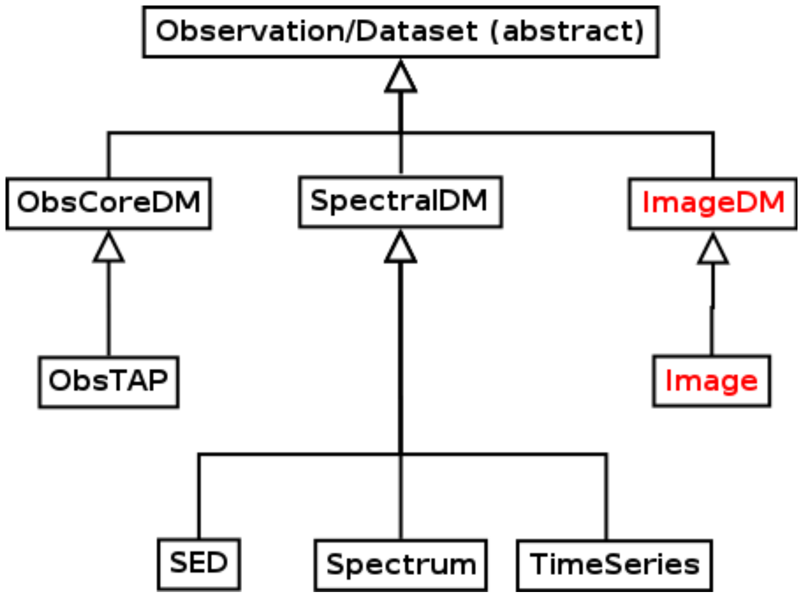


Figure 1. Role of the ImageDM in the VO data model architecture.

The role of the Image data model in the virtual observatory data model architecture is shown in Figure 1. The ImageDM is comparable to the Spectral data model, the core model for general spectrophotometric sequences, used to model spectra, time series, and so forth. Much of the metadata is common between the two, and also common with the ObsCoreDM that is used to model generic datasets of any type. All of these derive from a common, abstract Observation data model, used to model generic datasets of any type.

The key difference between the Image and Spectral data models is that, while the ImageDM is based upon a core model comprised of a regularly sampled n-D array with a separable WCS, the Spectral data model describes irregularly spaced spectrophotometric sequences where the world coordinates, bin size, errors, and so forth may be described independently for each sample in the sequence (Spectral does not use a separable WCS). The ImageDM is more efficient and easy to use for large data arrays, while the SpectralDM provides a more complete description of the samples in a spectrophotometric sequence. The underlying physics of the observation is much the same in both cases.

2.1 Image Data Model Architecture

The Image data model architecture is illustrated in Figure 2.

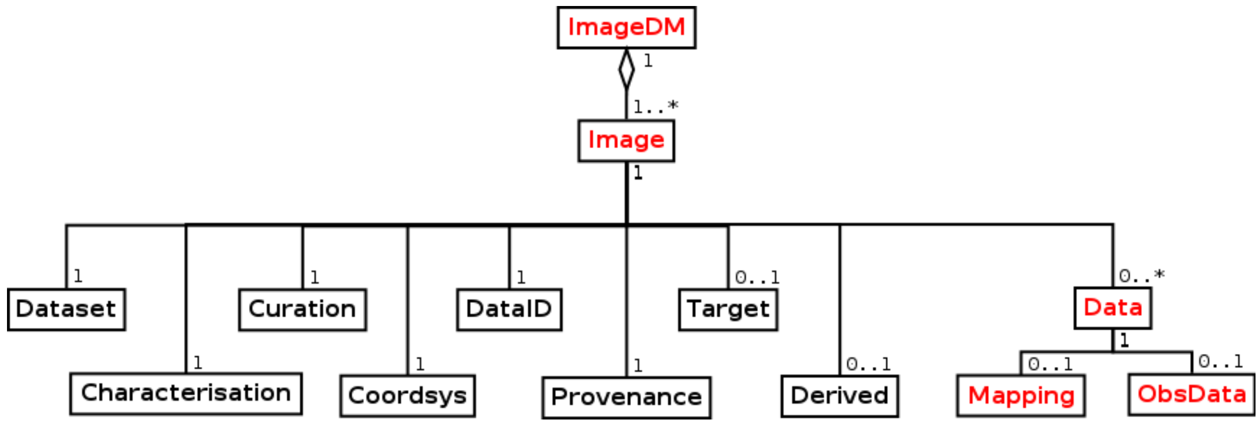


Figure 2. The Image data model architecture and classes.

An aggregate image dataset, or set of related Image dataset instances, consists of one or more Image instances (aggregate image datasets occur for example in discovery queries). Each Image instance consists of some standard metadata elements, possibly augmented by custom extension metadata, and zero or more Data elements. The Data element contains the image data (n-D array of data samples) and any associated Data-specific metadata.

2.2 Use Case Examples

The following sections illustrate which elements of the ImageDM are used in each of the primary use cases, including those identified in section 1.2. [*FITS serialization examples are included below for illustration and to aid the design effort, but would not necessarily be included here in the final ImageDM specification.*]

2.2.1 Query Response

In a query response (or any other case where an image dataset is referenced and described), we have a single Image instance with no Data element. An *Access* element in the query response (not shown in Figure 2) contains an access URL that may be used to retrieve the referenced image dataset. Since the Mapping (WCS) element is part of the Data element it is normally excluded from the query response, but can be retrieved for a given image dataset by an additional query.

[*I think this will work fine for our data access use cases such as SIAV2. A Mapping instance contains arrays that can be large, and is more detailed than needed for most discovery use cases, so retrieving Mapping with a separate query or access reference is reasonable. In the case of multiple subarrays where there are multiple Data elements it would not be practical to include metadata from the Data elements directly in the query response in any case. How metadata such as Mapping are to be retrieved from Data is part of the access protocol, not the ImageDM, but probably a simple access URL for this purpose, returning a table wherein each row describes a Data element, would suffice.*]

2.2.2 Simple Image

The simple image has a single Image instance containing metadata and a single Data element. [*The FITS serialization is a simple FITS image.]*

2.2.3 Single Sparse Image

A single sparse image has a single Image instance containing metadata and a single Data element; the only difference from the simple image is in Mapping, which is used to describe the WCS attributes of the valid data samples. [*The FITS serialization is a FITS MEF consisting of an image plus a BINTABLE extension containing data for a TAB WCS.]*

2.2.4 Multiple Subarray Image

Use case **3a** consists of a single Image instance with multiple subarrays, each in a separate Data element. The single Image instance contains metadata for the entire image dataset. Each Data element contains an n-D image array with associated metadata including Mapping and ObsData, both of which are optional elements. [*The FITS serialization is a FITS MEF consisting of a dataless primary header unit (PHU), followed by a series of FITS image extensions, one per Data element. The image extensions need contain only minimal standard FITS metadata since most observational metadata is in the PHU. If a Data element is sparse then an additional BINTABLE extension is needed to contain the TAB WCS for that element.*]

Use case **3b** is an aggregation of related but otherwise independent Image instances. This is a type of *complex data* aggregate wherein standard dataset instances are combined to model more complex data. How the Image instances are related is defined by the application and is not part of the ImageDM. The Image instances may be simple, sparse, or may contain multiple subarrays; any legal Image instance is permitted. [*The FITS serialization for this case would not normally be a single FITS file, but rather multiple separate files, however a MEF aggregate could be used.*]

2.3 Observational Data Pass-Through

Any ImageDM Data element may optionally contain a reference (via the optional *ObsData* element) to the external observational or instrumental data used to create the described

Image dataset. Given a sufficiently advanced application, this could be used for example to do analysis on event data directly in the event domain, possibly doing multiwavelength analysis combining event data with image data from other domains (likewise for visibility data although this is less likely to be useful due to the complexity and data volumes associated with visibility data). Generic analysis applications would ignore the ObsData element and work only with the provided already-imaged view of the data.

The ObsData element contains a subset of the *ObsCoreDM* (Observation core components data model), and is capable of describing any science data product, given defined values for fields such as the data product type, subtype (domain-specific), and format. The values given should be the same as would be provided by an ObsTAP service describing the same data products.

The image dataset described by an Image instance may be *virtual data*, generated on demand when accessed. The data described by the corresponding ObsData instance may likewise be virtual data. In particular, ideally the data returned by an ObsData access reference should be filtered to correspond to the same multi-dimensional region of space as the described Image dataset. If this is not possible, the observational data may be a superset of the data used to produce the Image instance.

2.4 Sparse Data

The strategies for dealing with sparse data have already been introduced in the sections above. Two techniques are provided to directly deal with sparse data via the ImageDM: a single sparse image, and an overall image dataset space containing multiple subarrays. Both techniques may be combined within the same Image dataset.

In the case of a **single sparse image**, the approach is to 1) include in the data array only the actual data samples (flux values or other derived quantities) for each image axis that is sparse, and 2) define a *Mapping* to associate, for each sparse axis, WCS values for each data sample provided. The affected axes of the Image data array thus become a simple pixel list, or alternatively a list of regions where data has been sampled, as opposed to a fully populated n-D data array. Implementing this requires that Mapping, for each Image axis which is sparse, be able to explicitly map data samples, or ranges of samples, to WCS values. [*The TAB coordinate type in FITS provides a standard mechanism for this and should be referenced here as the standard mechanism to do this*. *Coordinate and index vectors are included in Mapping to implement this feature for the ImageDM. Note that for the ImageDM, the TAB arrays can be included directly in Mapping (since the ImageDM can support embedded arrays), whereas FITS requires a separate BINTABLE extension to store the TAB coordinate and optionally index data*.]

A completely different approach to dealing with individual sparse images is to use image compression – with a good compression algorithm, unobserved or constant regions of a fully sampled image array should compress to essentially nothing. Compression is a feature of the specific image serialization used, and not explicitly part of the ImageDM. [*We should say more about this, at least in the discussion of serializations later in the document*.]

The second approach to modeling sparse images is to compose the image using **multiple subarrays**. Common examples where this is useful might be a wide-field multi-CCD detector with gaps between the CCDs (due to misalignment of the CCDs each may require a separate WCS), or a multi-band image wherein the images are spatially registered but coverage is widely spaced along the spectral axis (this case is frequently encountered for both O/IR and radio data).

Multiple subarrays with shared dataset metadata are directly addressed by the ImageDM. An Image instance may contain any number of subarrays, so long as they do not overlap in the full multi-dimensional space of the image (e.g., in the case of a spectral cube where the spectral axis is sparse, data for successive spectral bands do not overlap since they are distinct, non-overlapping spectral bands, even though the bands may share the same spatial region).

A possibly surprising aspect of multiple subarray data is that the overall Image instance, while it has well defined dimensionality and coverage, may not have an overall image geometry or WCS since the subarrays need not be co-aligned to the same pixel grid. It is not an n-D image in the conventional sense, but nonetheless it is "image data" since it is composed of individual image subarrays (and can be represented by the ImageDM).

2.4.1 Modeling Event Data as Sparse Data

While pass-through of event data via *ObsData* is probably the best way to expose the underlying event data used to image an event dataset (external non-VO standards already exist to represent event data), it is possible to use the single sparse image formalism to directly represent event data via the ImageDM.

What the ImageDM fundamentally defines is an N-dimensional data array with an associated M-dimensional WCS, where N and M do not need to be equal. In particular, the WCS dimensionality M may exceed the data array dimensionality N. Instrumental event data consist of a flux value (PHA or pulse height) at a given time and detector coordinate. The instrumental flux measure (e.g. PHA) is typically calibrated and converted into energy, with the fundamental elements of an event list being the event time of detection, the detector coordinates of the event, the corresponding celestial coordinates, and the calibrated energy or instrumental PHA value.

In terms of the ImageDM, all measureable physical attributes of an event are sparse: the two spatial coordinates, time, and (if calibrated) energy. Hence an event list may be directly represented in the ImageDM, where the data array is a photon list with time and the two spatial coordinates as the independent variables (image dimensions), and PHA as the observable. Alternatively if PHA is calibrated and converted to energy, the event list may be completely represented as a 4-dimensional WCS consisting of time, the two spatial coordinates (e.g., RA, DEC), and energy, with the observable being a constant 1 count per event (hence the numerical data array can actually be eliminated). Another way of thinking about this is that an event list is very similar to a sparse 4-D cube, with one photon per voxel.

While representation of an event list as a sparse image is possible and is an interesting limiting case for the ImageDM, pass-through of event data in a standard format already widely adopted by the high energy community is likely to be the more practical approach.

3 Data Model Classes

3.1 Types of Metadata

Metadata to describe an image instance are grouped into a number of component data models as summarized in the table below, and are explained in more detail in the sections that follow.

Generic Dataset Metadata	
Dataset	General metadata describing the overall dataset
Dataset.Image	Metadata specific to an Image dataset
DataID	Dataset identification (creation)
Provenance	Instrumental or software provenance
Curation	Publisher metadata
Target	Observed target, if any
CoordSys	Coordinate system frames
Char	Dataset characterization
Characterization Metadata	
Char.SpatialAxis	Spatial measurement axis
Char.SpectralAxis	Spectral measurement axis, e.g., wavelength
Char.TimeAxis	Temporal measurement axis
Char.Polarization	Polarization Axis
Char.FluxAxis	Observable, normally a flux measurement
Char.*.Coverage	Coverage in any axis
Char.*.Resolution	Resolution on any axis
Char.*.SamplingPrecision	Sampling or Precision on any axis
Char.*.Accuracy	Accuracy and error in any axis
Data Element	
General attributes, geometry, data	Included at the root of the Data element
Mapping	World coordinate system for data array
ObsData	Reference to external observational data

Generic dataset metadata is general metadata used to describe any type of VO dataset. The **Dataset.Image** element contains high level, image-specific metadata used to describe the overall image dataset (e.g., the number of subarrays, image geometry, number of pixels on each axis, and so forth). **Characterization metadata** physically characterizes the dataset in terms of the spatial, spectral, temporal, and polarization measurement axes as well as the observable. Characterization is also generic dataset metadata but is broken out separately in the table above to show the major elements of the characterization model. A **Data Element** includes metadata describing the element itself as well as (optionally) the actual data array. Most image metadata is generic dataset metadata that can be used to describe any type of data, and is shared with other VO data models.

An Image instance may contain zero, one, or multiple Data elements. Data element metadata optionally includes a **Mapping Element** defining the WCS for the specific Data element, as well as an optional **ObsData Element** referencing the observational data corresponding to the Data element data array (information may be lost when imaging a dataset, and the optional ObsData element can provide an alternate, more fundamental view of the data).

Each of these types of query response metadata is discussed in more detail in the sections that follow.

[*In what follows, for this first draft of the ImageDM we merely summarize the data model classes, highlighting what is new. Most of the generic dataset metadata including characterisation is already described in other VO documents. A later version of this document will more fully describe each data model class.*]

3.2 Generic Dataset Metadata

The following metadata components (with the exception of *Dataset.Image*) are shared with other VO data models such as ObsCore and the Spectral data model.

3.2.1 Dataset Metadata

General dataset metadata describes the overall dataset. In the case of the ImageDM, *Dataset* is extended to add some image-specific dataset metadata. In what follows, mandatory elements of the ImageDM are indicated as "MAN", and optional elements as "OPT", however an application that uses the ImageDM may redefine what is mandatory and optional, or may impose additional constraints on use of the data model.

UTYPE	*Description*	*Req*	*Default*
Dataset.DataModel.Name	Data model name and version	MAN	Image-2.0
Dataset.DataModel.Prefix	Data model prefix	MAN	im
Dataset.DataModel.URL	Reference URL for the data model	OPT	
Dataset.Type	Type of VO dataset	MAN	image
Dataset.Subtype	Type of data product (archive-specific)	OPT	
Dataset.CalibLevel	Calibration level	MAN	
Dataset.Length	Total number of voxels in image dataset	MAN	
Dataset.Image.Nsubarrays	Number of image subarrays	MAN	
Dataset.Image.Naxes	Number of physical image axes	MAN	
Dataset.Image.Naxis	Length of each image axis	MAN	
Dataset.Image.Pixtype	Pixel datatype	MAN	
Dataset.Image.WCSAxes	Enumeration of the WCS axes types	MAN	
Dataset.Image.DataRef	Reference URL for Data element metadata	OPT	

Dataset.DataModel.Name is a string identifying the data model used for the current dataset instance. For ImageDM-compliant data this should be the string "Image-2.0". For the ImageDM the data model prefix (e.g., for use to compose Utypes) is "im". The data model reference URL is not currently used, but is reserved for use with a future version of the data model mechanism, to reference a machine-readable definition of the data model.

For an ImageDM instance, ***Dataset.Type*** must be "image". A subtype may optionally be given to specify the image data product type in terms of a specific archive or data collection (***Dataset.Subtype*** is semantically equivalent to *Obs.DataProductSubtype* in the ObsCoreDM [ObsTAP-XX], but is restricted to image datasets whereas ObsCore describes general data products or observations).

Dataset.CalibLevel defines the calibration level [ObsTAP-XX] of the dataset. ***Dataset.Length*** specifies the total number of voxels (data samples) in the full image dataset. If the image is sparse this may be less than is implied by the product of the image axis lengths.

3.2.1.1 Dataset.Image Metadata

The ***Dataset.Image*** subgroup contains image-specific metadata describing the overall image dataset. ***Nsubarrays*** specifies the number of image subarrays (Data elements) in the Image instance. This is zero in the case of a dataless (metadata only) Image instance, one for a simple image, and *N* for an image consisting of N subarrays.

Naxes specifies the physical number of image axes, i.e., the image dimensionality; note that the WCS dimensionality may exceed that of the physical image. In the case of an image with multiple subarrays, Naxes refers to the dimensionality of the overall image dataset. ***Naxis*** is a 2-dimensional array specifying the length (number of voxels) of each image axis of each subarray. Each row specifies the axis lengths for a single subarray. ***Pixtype*** specifies the numeric pixel or voxel datatype as in VOTable [VOTable-XX].

Naxis example: 500 500 70 1

WCSAxes is an array of strings, wherein each successive element of the array specifies the world coordinate type of the corresponding WCS axis. The array length of WCSAxes is the dimensionality of the WCS. If WCSAxes is not specified or has zero length this indicates that the image has no associated WCS. If the image has a WCS then WCSAxes must be provided, otherwise it may be omitted.

WCSAxes example: RA DEC FREQ STOKES

An image ***Data element*** may or may not be included in an instance of the ImageDM, e.g., when the ImageDM instance is used to describe an image dataset any Data elements are excluded. In such a case a ***DataRef*** attribute may be provided to retrieve Data element metadata. The value is a URL that is used to reference or retrieve data element metadata for all image subarrays. The details of how this is done are specified in the data access protocol. [*This makes me wonder if possibly the whole Dataset.Image element should be defined by a data access protocol such as SIAV2 rather than in the ImageDM, but we will leave it here for now. What DataRef will return is a VOTable, wherein each table row contains the metadata (minus the Data.values attribute) for a single image subarray. For a simple image there is only a single table row. Note that the image data is returned in the image dataset that is retrieved by the main access reference returned in a query response.*]

3.2.2 Dataset Identification Metadata

Dataset identification metadata is used to describe the fundamental identity of a dataset, including the data collection it belongs to and how the dataset was created.

UTYPE	Description	Req	Default
DataID.Title	Dataset title (brief description)	MAN	
DataID.Creator	Creator name (string)	REC	
DataID.Collection	IVOA Identifier of collection	REC	
DataID.DatasetID	IVOA Dataset ID	OPT	
DataID.CreatorDID	Creator assigned dataset identifier	OPT	
DataID.Date	Data processing/creation date	OPT	
DataID.Version	Version of creator-produced dataset	OPT	
DataID.CreationType	Dataset creation type	OPT	archival
DataID.Logo	URL for creator logo	OPT	
DataID.Contributor	Contributor name	OPT	

DataID.Title is a short, human-readable description of a dataset, and should be less than one line of text. Information such as the instrument or survey name, observing mode or intent, filter, target name, etc., is typically included in a condensed form. The contents of DataID.Title are up to the data provider. ***DataID.Creator*** identifies the entity that created the dataset, and should be a short string consistent with the RSM specification, e.g., "NRAO". ***DataID.Collection*** is the registered IVOA identifier of the data collection to which the dataset belongs, e.g., "ivo://nrao/vla".

A dataset identifier is a URI used to uniquely identify a dataset within some well-defined context [REF]. ***DataID.DatasetID*** is some well-known, IVOA-recognized identifier for the dataset, for example an ADS dataset identifier. ***CreatorDID*** is an IVOA dataset identifier (if any) assigned by the entity which created the dataset *content*, typically (but not always) an observatory or survey project. If the dataset referred to is virtual data, CreatorDID refers to the parent dataset from which the virtual data will be created. If a CreatorDID has been assigned to a dataset it should be provided, otherwise it should be omitted. ***DataID.Date***, specified in ISO time format, specifies the date when the dataset was created or last modified by the DataID.Creator entity. If a dataset is modified or replaced without changing its CreatorDID, DataID.Date and ***DataID.Version*** should be updated accordingly. ***DataID.CreationType*** describes how the dataset returned by the service was or will be created from the original data.

3.2.3 Provenance Metadata

Provenance metadata are used to provide information on the scientific origin of the dataset, from either the observing or the processing point of view.

UTYPE	Description	Req	Default
Provenance.ObsConfig.Instrument	Instrument name	REC	
Provenance.ObsConfig.Bandpass	Bandpass name, e.g., filter	OPT	
Provenance.ObsConfig.DataSource	Original source of data	REC	survey

Provenance.Proposal.Identifier	ID of associated proposal if any	OPT	

Provenance.ObsConfig.Instrument is a short string identifying the instrument used to create the data (instrument may be an actual telescope instrument or something else, e.g., a program in the case of theory data). ***Provenance.ObsConfig.Bandpass*** is a short string specifying the bandpass name if any, e.g., a filter name or an instrumental bandpass such as K, Q, and so forth. Values specified with Provenance.ObsConfig.Bandpass may be used as input to a parameter such as BAND in a data access protocol to refine a query (if this feature is supported by the service).

Provenance.ObsConfig.DataSource describes the original source of the data. Valid values include “survey”, “pointed”, “custom”, “theory”, and “artificial” [*need to expand upon this*].

Provenance.Proposal.Identifier specifies the proposal ID of the proposal (if any) associated with the dataset. This might be used for example, to associate observed and derived data products within an archive, that are all connected to the same observing program.

3.2.4 Curation Metadata

Curation metadata describes who curates the dataset and how it is published to the VO.

UTYPE	*Description*	*Req*	*Default*
Curation.Publisher	Publisher	REC	
Curation.PublisherID	URI for publishing entity	OPT	
Curation.PublisherDID	Publisher's ID for the dataset	REC	
Curation.ReleaseDate	Date curated dataset last modified	OPT	
Curation.Version	Publisher’s version of the dataset	OPT	
Curation.Rights	Restrictions if any on data access	OPT	
Curation.Reference	URL or Bibcode for documentation	OPT	
Curation.Contact.Name	Contact name	OPT	
Curation.Contact.Email	Contact email	OPT	

Curation.Publisher is a short string naming the publisher of the data, e.g., a data archive or data center, or an indexing service such as the ADS. ***Curation.PublisherID*** is the URI of the publisher as registered within the VO. ***Curation.PublisherDID*** is the IVOA dataset identifier (URI) assigned by the publisher to identify the dataset within its holdings. ***Curation.ReleaseDate*** is the date (ISO format) when the dataset was or will be publically released (hence metadata for proprietary datasets can be available prior to the release of the actual dataset). ***Curation.Version*** indicates the publisher’s version of the dataset, and should be updated if the publisher modifies a dataset. Curation.ReleaseDate and Curation.Version refer to the dataset *as curated by the publisher*, hence can differ from the same values given in DataID, which refer to the *content* of the dataset as generated by the dataset Creator.

Curation.Reference is a forward link to publications that reference the dataset; multiple instances are permitted. ***Curation.Rights*** specifies whether the dataset is "public" or "proprietary". Proprietary data requires authentication and authorization by the data provider to access, and once downloaded should be protected from subsequent access on the client side. ***Curation.Contact.Email*** and ***Curation.Contact.Name*** indicate the person or group responsible for curating the data.

3.2.5 Astronomical Target Metadata

Target metadata describes the astronomical target observed, if any. The attributes given are typically known apriori, and are not derived from the data. Image datasets that do not reflect the analysis of a single target should omit most of this metadata, with the exception of Target.Name in the case of a pointed observation of a specific target.

UTYPE	Description	Req	Default
Target.Name	Target name	REC	
Target.Description	Target description.	OPT	
Target.Class	Target or object class	OPT	
Target.SpectralClass	Target or object spectral class	OPT	
Target.Pos	Target position	OPT	
Target.Redshift	Target redshift	OPT	
Target.VarAmpl	Target variability amplitude, typical	OPT	

Target.Name is a short string identifying the observed astronomical object, suitable for input to a name resolver. ***Target.Description*** should provide a brief description of the target object. ***Target.Class*** is the object class if known, e.g., star, galaxy, agn, qso, and so on [*at present we do not know of any controlled vocabulary that can be referenced here*]. ***Target.SpectralClass*** and ***Target.Redshift*** record the spectral class and redshift of the target if known. ***Target.VarAmpl*** specifies the variability of the target as a value in the range 0.0 to 1.0.

3.2.6 Derived Metadata

Derived metadata is metadata pertaining to the observed target that is derived by analysis of the current dataset.

UTYPE	Description	Req	Default
Derived.SNR	Signal-to-noise of observed target	OPT	
Derived.Redshift.Value	Measured redshift for target	OPT	
Derived.Redshift.StatError	Statistical error for measured redshift	OPT	
Derived.Redshift.Confidence	Confidence value for redshift	OPT	
Derived.VarAmpl	Variability amplitude as fraction of mean	OPT	

For many image datasets this generic metadata is not relevant and may be omitted, but if a central target is analyzed derived attributes may be specified.

3.2.7 Coordinate System Metadata

Coordinate system metadata describes the coordinate system reference frames used within the ImageDM instance, including Characterization.

Utype	*Description*	*REQ*	*Default*

CoordSys.ID	ID string for coordinate system	OPT	
CoordSys.SpaceFrame			
CoordSys.SpaceFrame.ID	ID string for spatial frame	OPT	
CoordSys.SpaceFrame.Name	Spatial coordinate frame name	OPT	ICRS
CoordSys.SpaceFrame.UCD	Space frame UCD	OPT	
CoordSys.SpaceFrame.RefPos	Origin of SpaceFrame	OPT	UNKNOWN
CoordSys.SpaceFrame.Equinox	Equinox	OPT	2000.0
CoordSys.TimeFrame			
CoordSys.TimeFrame.ID	ID string for time frame	OPT	
CoordSys.TimeFrame.Name	Timescale	OPT	TT
CoordSys.TimeFrame.UCD	Time frame UCD	OPT	
CoordSys.TimeFrame.RefPos	Location for times of photon arrival	OPT	TOPOCENTER
CoordSys.TimeFrame.Zero	Zero point of timescale in MJD	OPT	0.0
CoordSys.SpectralFrame			
CoordSys.SpectralFrame.ID	ID string for spectral frame	OPT	
CoordSys.SpectralFrame.Name	Spectral frame name	OPT	
CoordSys.SpectralFrame.UCD	Spectral frame UCD	OPT	
CoordSys.SpectralFrame.RefPos	Spectral frame origin	OPT	TOPOCENTER
CoordSys.SpectralFrame.Redshift	Redshift value used if restframe corrected	OPT	0.0
CoordSys.RedshiftFrame			
CoordSys.RedshiftFrame.ID	ID string for redshift frame	OPT	
CoordSys.RedshiftFrame.Name	Redshift frame name	OPT	
CoordSys.RedshiftFrame.UCD	Redshift frame UCD	OPT	
CoordSys.RedshiftFrame.RefPos	Redshift frame origin	OPT	UNKNOWN
CoordSys.RedshiftFrame.DopplerDefinition	Type of redshift	OPT	UNKNOWN
CoordSys.FluxFrame			
CoordSys.FluxFrame.ID	ID string for flux frame	OPT	
CoordSys.FluxFrame.Name	Name of photometric band	OPT	
CoordSys.FluxFrame.UCD	UCD for photometric calibration	OPT	phot.mag
CoordSys.FluxFrame.refID	URI for photometric calibration	OPT	

These reference frames apply to all spatial, spectral, time, and photometric coordinates used in the ImageDM instance (including Characterization) unless otherwise specified. An explicit polarization coordinate frame is omitted since polarization states are enumerated and no coordinate system is required. Note that spatial coordinates are not limited to the celestial sphere; any spatial coordinate frame specified in the data model may be specified, including solar and planetary coordinate systems, although the default is ICRS.

3.3 Dataset Characterization Metadata

The Characterization data model [REF] provides a standard, generic data model to describe the characteristics of a dataset. The coverage, sampling, and data quality of the dataset are uniformly characterized for the spatial, spectral, time, and polarization measurement axes, as well as for the observable.

The Characterization data model (CharDM, Char) is directly incorporated into the ImageDM. We do not attempt to fully describe the elements of the CharDM in this document. The subset of the CharDM used by the ImageDM is briefly summarized, and any usage specific to image data is noted. The CharDM [REF] and ObsTAP (ObsCoreDM) [REF] documents should be referred to for the details.

The *CoordSys* element of the ImageDM defines the coordinate systems used in Char as well as elsewhere within Image. While the full CharDM includes its own coordinate system definition, for simplicity coordinate systems are required to be uniform throughout the ImageDM metadata including Char (the Data element including Mapping is handled differently as we will see in section 3.4).

Additional elements of the CharDM are permissible but are not required, and clients should not assume any additional metadata beyond the mandatory elements is provided.

3.3.1 Spatial Axis Characterization

The element ***Char.SpatialAxis*** is used to characterize the spatial axis of the dataset.

Utype	*Description*	*REQ*	*Default*
Char.SpatialAxis.Name	Name for spatial axis	OPT	
Char.SpatialAxis.UCD	UCD for spatial coord	OPT	
Char.SpatialAxis.Unit	Unit for spatial coord	OPT	
Char.SpatialAxis.Coverage.Location.Coord	Spatial Position	MAN	
Char.SpatialAxis.Coverage.Bounds.Extent	Aperture angular size	OPT	
Char.SpatialAxis.Coverage.Bounds.Limits. LoLimit2Vec	Lower bounds of image spatial coordinates	MAN	
Char.SpatialAxis.Coverage.Bounds.Limits. HiLimit2Vec	Higher bounds of image spatial coordinates	MAN	
Char.SpatialAxis.Coverage.Support.Area	Aperture region	OPT	
Char.SpatialAxis.Coverage.Support.Extent	Field of view area	OPT	
Char.SpatialAxis.SamplingPrecision.SampleExtent	Spatial bin size	OPT	
Char.SpatialAxis.SamplingPrecision.SamplingPrecisionRefVal.FillFactor	Spatial sampling filling factor	REC	
Char.SpatialAxis.Accuracy.StatError	Astrometric statistical error	OPT	
Char.SpatialAxis.Accuracy.SysError	Astrometric systematic error	OPT	
Char.SpatialAxis.CalibrationStatus	Type of spatial coord calibration	REC	
Char.SpatialAxis.Resolution.RefVal	Spatial resolution of data	REC	

The central coordinates (*Location*) as well as the corners of the image footprint (*Bounds.Limits*) of the spatial coverage of the data are mandatory metadata, except for data where spatial location is not applicable, e.g., theory or artificial data. If possible the filling factor and spatial resolution should also be characterized. The filling factor is the fraction (0.0 to 1.0) of the spatial image footprint that has valid data samples. Images that are spatially sparse will have a spatial filling factor less than 1.0. Additional metadata should be provided where possible. The presence of a spatial calibration (WCS) should also be indicated.

3.3.2 Spectral Axis Characterization

The element ***Char.SpectralAxis*** is used to characterize the spectral axis of the dataset.

Utype	Description	REQ	Default
Char.SpectralAxis.Name	Name for spectral axis	OPT	
Char.SpectralAxis.UCD	UCD for spectral coordinate	REC	
Char.SpectralAxis.Unit	Unit for spectral coordinate	OPT	
Char.SpectralAxis.Coverage.Location.Coord	Spectral coordinate value	OPT	
Char.SpectralAxis.Coverage.Bounds.Extent	Width of spectral coverage	OPT	
Char.SpectralAxis.Coverage.Bounds.Limits.LoLimit	Start in spectral coordinate	MAN	
Char.SpectralAxis.Coverage.Bounds.Limits.HiLimit	Stop in spectral coordinate	MAN	
Char.SpectralAxis.Coverage.Support.Extent	Effective width of spectrum	OPT	
Char.SpectralAxis.SamplingPrecision.SampleExtent	Wavelength bin size	OPT	
Char.SpectralAxis.SamplingPrecision.SamplingPrecisionRefVal.FillFactor	Spectral sampling filling factor	OPT	
Char.SpectralAxis.Accuracy.BinSize	Spectral coord bin size	OPT	
Char.SpectralAxis.Accuracy.StatError	Spectral coord statistical error	OPT	
Char.SpectralAxis.Accuracy.SysError	Spectral coord systematic error	OPT	
Char.SpectralAxis.CalibrationStatus	Type of spectral coord calibration	OPT	
Char.SpectralAxis.Resolution.RefVal	Spectral resolution FWHM	REC	
Char.SpectralAxis.ResolPower.RefVal	Spectral resolving power	REC	

The bounds of the spectral coverage of the dataset (*Bounds.Limits*) are mandatory metadata to specify the spectral coverage of the dataset. If possible the UCD for the spectral coordinate (specifying the physical type of spectral coordinate), and the spectral resolution and resolving power should also be specified. Additional metadata should be provided where possible.

3.3.3 Time Axis Characterization

The element ***Char.TimeAxis*** is used to characterize the time axis of the dataset.

Utype	Description	REQ	Default
Char.TimeAxis.Name	Name for time axis	OPT	
Char.TimeAxis.UCD	UCD for time	OPT	
Char.TimeAxis.Unit	Unit for time	OPT	
Char.TimeAxis.Coverage.Location.Coord	Midpoint of exposure on MJD scale	OPT	
Char.TimeAxis.Coverage.Bounds.Extent	Total exposure time	OPT	
Char.TimeAxis.Coverage.Bounds.Limits.LoLimit	Start time	MAN	
Char.TimeAxis.Coverage.Bounds.Limits.HiLimit	Stop time	MAN	
Char.TimeAxis.Coverage.Support.Extent	Effective exposure time	OPT	
Char.TimeAxis.SamplingPrecision.SampleExtent	Time bin size	OPT	
Char.TimeAxis.SamplingPrecision.SamplingPrecisionRefVal.FillFactor	Time sampling filling factor	OPT	
Char.TimeAxis.Accuracy.BinSize	Time bin size	OPT	
Char.TimeAxis.Accuracy.StatError	Time coord statistical error	OPT	
Char.TimeAxis.Accuracy.SysError	Time coord systematic error	OPT	
Char.TimeAxis.CalibrationStatus	Type of coord calibration	OPT	

Char.TimeAxis.Resolution.RefVal	Time resolution FWHM	OPT	

Specification of the bounds of the time coverage of the dataset (*Bounds.Limits*) is mandatory. Additional metadata should be specified if possible, especially if time coverage is an important attribute of the dataset.

3.3.4 Polarization Axis Characterization

The element ***Char.PolAxis*** is used to characterize the polarization axis of the dataset.

Utype	*Description*	*REQ*	*Default*
Char.PolAxis.Name	Name for polarization axis	OPT	
Char.PolAxis.UCD	UCD for polarization type	OPT	
Char.PolAxis.Enumeration	List of polarization states present	REC	

Characterization of the polarization axis is optional, *unless* the dataset measures polarization, in which case it is mandatory. At a minimum the type of polarization measured (*PolAxis.UCD*) and a listing of the polarization states measured (*PolAxis.Enumeration*) should be provided for datasets that measure polarization.

The following names are reserved for the Stokes parameters, and for left and right-handed circular polarization:

POL Value	***Description***
I	Stokes I (total intensity).
Q, U	Stokes Q and U (linear polarization).
V	Stokes V (circular polarization).
L, R	Left- and right-handed circular polarization.

3.3.5 Flux Characterization

The element ***Char.FluxAxis*** is used to characterize the flux (observable) axis of the dataset.

Utype	*Description*	*REQ*	*Default*
Char.FluxAxis.Name	Name for flux axis	OPT	
Char.FluxAxis.UCD	UCD for flux	MAN	
Char.FluxAxis.Unit	Unit for flux	REC	
Char.FluxAxis.Accuracy.StatError	Flux statistical error	OPT	
Char.FluxAxis.Accuracy.SysError	Flux systematic error	OPT	
Char.FluxAxis.CalibrationStatus	Type of flux calibration	REC	

Specification of the UCD for flux values is mandatory to indicate the physical type of flux measure provided. The flux unit and calibration status should also be indicated. In particular the flux calibration status is important to indicate for applications such as SED generation that may require absolute flux calibration.

3.4 Data Element

Data element metadata describes a Data element of the image. In the case of an Image dataset instance the data values are included as well.

UTYPE	Description	REQ	Default
Data.ID	Unique identifier for the Data element	OPT	
Data.Naxes	Number of image axes	MAN	
Data.Naxis	Length of each axis in pixels	MAN	
Data.Pixtype	Pixel / voxel datatype	MAN	
Data.Encoding	Encoding used for the array data	OPT	"FITS"
Data.Length	Array length in voxels (actual count if sparse)	MAN	
Data.Size	Data array element size in bytes	MAN	
Data.Values	Array data	OPT	
Data.Mapping.*	World coordinate system	OPT	
Data.ObsData.*	Reference to original observational data	OPT	

Data.ID should be provided to uniquely identify a Data element if the image contains multiple Data elements. ***Data.Naxes*** and ***Data.Naxis*** specify the image geometry. ***Data.Pixtype*** specifies the pixel datatype as in VOTable [*should elaborate this.*]. ***Data.Encoding*** specifies the encoding of the array data, for example, the order of rows and columns of data in the array, or whether data is stored in n-D blocks (often this will be specified by the serialization, but not in all cases). ***Data.Length*** specifies the number of voxels (data samples) in the full image dataset; for a sparse image this may be less than the product of the image axis lengths. ***Data.Size*** specifies the size in bytes of the image data array. For an actual image dataset, ***Data.Values*** contains the actual array data.

The optional ***Data.Mapping*** and ***Data.ObsData*** elements are described separately below. Data.Mapping defines the mapping of the image axes to a world coordinate system. Data.ObsData optionally may be used to reference the more fundamental observational data used to create the image (the image array is essentially a standard "view" of this more fundamental data).

3.4.1 Mapping Metadata

The ***Mapping*** model specifies the physical image matrix and the transformation from image pixel coordinates to the specified world coordinate system (WCS). Images with any combination of spatial, spectral, time, polarization, or generalized linear axes are supported.

The Mapping model is compatible with FITS WCS, with some extensions to provide more generalized support for a WCS time axis, and for polarization. In most cases the elements of a FITS WCS may be directly converted to or from the Mapping representation. Conversion to and from a STC representation is also possible for most analytical projections. A Mapping instance is a fully encapsulated object and may be re-used in different contexts.

UTYPE	Description	REQ	Default

Image Axis to Intermediate Coordinate Transform			
Mapping.WCSNaxes	Number of WCS axes		
Mapping.WCSName	Name of the overall WCS		
Mapping.RefPixel	Reference pixel		
Mapping.RefValue	WCS value at reference pixel		
Mapping.CDMatrix	Coord definition matrix		
Mapping.PCMatrix	Coord definition matrix		
Mapping.CDelt	World coord delta per pixel		
Mapping.AxisMap	Image-to-WCS axis mapping		
World Coordinate Transform			
Mapping.Axis			
Mapping.Axis.CoordType	Coordinate type		
Mapping.Axis.Unit	Coordinate unit		
Mapping.Axis.Name	Axis name		
Mapping.Axis.CoordValue	Table of explicit coordinate values		
Mapping.Axis.CoordIndex	Index into CoordValue		
Mapping.SpatialAxis			
Mapping.SpatialAxis.CoordType	Coordinate type as in FITS		
Mapping.SpatialAxis.Algorithm	Celestial projection		
Mapping.SpatialAxis.CoordFrame	Spatial coordinate frame		
Mapping.SpatialAxis.CoordEquinox	Coordinate equinox (if used)		
Mapping.SpatialAxis.Unit	Unit for coordinate value		
Mapping.SpatialAxis.Name	Axis name (optional)		
Mapping.SpatialAxis.PV	Optional parameters for transform		
Mapping.SpatialAxis.PS	Optional parameters for transform		
Mapping.SpatialAxis.LonPole	Native longitude of the celestial pole		
Mapping.SpatialAxis.LatPole	Native latitude of the celestial pole		
Mapping.SpatialAxis.CoordValue	Coordinate values		
Mapping.SpatialAxis.CoordIndex1	Coordinate index for first axis		
Mapping.SpatialAxis.CoordIndex2	Coordinate index for second axis		
Mapping.SpectralAxis			
Mapping.SpectralAxis.CoordType	Coordinate type as in FITS		
Mapping.SpectralAxis.Algorithm	Algorithm type as in FITS		
Mapping.SpectralAxis.RestFreq	Rest frequency of spectral line		
Mapping.SpectralAxis.RestWave	Rest wavelength of spectral line		
Mapping.SpectralAxis.CoordUnit	Unit for spectral coordinate value		
Mapping.SpectralAxis.CoordName	Axis name (optional)		
Mapping.SpectralAxis.PV	Optional parameters for transform		
Mapping.SpectralAxis.PS	Optional parameters for transform		
Mapping.SpectralAxis.CoordValue	Table of explicit spectral coord values		
Mapping.SpectralAxis.CoordIndex	Index into coordinate array		
Mapping.TimeAxis			
Mapping.TimeAxis.CoordType	Time scale		
Mapping.TimeAxis.RefPosition	Time reference position		
Mapping.TimeAxis.RefDirection	Time reference direction		
Mapping.TimeAxis.MJDRef	MJD time zero (for time offsets)		

Mapping.TimeAxis.CoordUnit	Time unit		
Mapping.TimeAxis.CoordName	Time axis name		
Mapping.TimeAxis.CoordValue	Table of explicit time coordinate values		
Mapping.TimeAxis.CoordIndex	Index into coordinate array		
Mapping.PolAxis			
Mapping.PolAxis.CoordType	Coordinate type as in FITS		
Mapping.PolAxis.CoordName	Polarization axis name		
Mapping.PolAxis.CoordValue	Polarization state at pixel index		

The WCS transformation consists of a general linear transformation of the input image pixel coordinates (with the transform represented either as the CD matrix or as the PC matrix plus CDELT), followed optionally by a nonlinear transformation to produce the final world coordinates. To apply the linear transformation one first subtracts the coordinates of the reference pixel, then applies the transformation matrix, and finally adds the world coordinates at the reference pixel to establish the zero point. The result is a linear transformation of the input pixel coordinates to "intermediate" (linear) world coordinates. An optional nonlinear transform may then be applied to get to the final world coordinates.

In the FITS representation the CTYPE keyword is used to specify both the coordinate type and the nonlinear algorithm if any to be applied to the axis. In Mapping these are broken out into separate ***CoordType*** and ***Algorithm*** attributes. Mapping uses array-valued attributes to represent arrays, whereas FITS uses multiple keywords, one for each array element. If multiple WCS instances are required for an image, this can be expressed by merely having multiple instances of Mapping, hence there is no need for the "a" suffix used in FITS WCS to express multiple coordinate systems.

In Mapping the ***AxisMap*** attribute is used to map the axes *of the intermediate world coordinates* (i.e., after the CD or PC transform, which may transpose or rotate the image axes) to the axes of the final world coordinate system. The spatial, spectral, time, or polarization transforms may then be applied independently to the associated intermediate world coordinate values without any further concern with image axes. A generalized linear transform ***Mapping.Axis*** is also provided to allow any type of linear transform to be applied to an axis.

A nonlinear coordinate system may be represented either as a continuous function consisting of a well-known projection or algorithm of some sort (e.g., TAN, F2V, etc.), or as a lookup table (via the optional ***CoordValue*** and ***CoordIndex*** arrays) wherein each pixel index on the axis is directly assigned a world coordinate. [*This is the TAB coordinate type mechanism used to represent sparse data*.].

[*More needs to be added here to fully specify this metadata, in particular the vector representation, allowable units, and allowable coordinate types and algorithms, following the FITS model, but this should suffice to demonstrate the approach.*]

3.4.2 ObsData Metadata

The optional ***ObsData*** element provides a pass-through mechanism to retrieve the more fundamental observational data corresponding to an image dataset.

UTYPE	Description	REQ	Default
ObsData.DataProductType	Primary generic data product type	OPT	
ObsData.DataProductSubtype	Data product collection-specific type	OPT	
ObsData.CalibLevel	Calibration level	OPT	
ObsData.Reference	URL used to access the dataset	OPT	
ObsData.Format	Content format of the dataset	OPT	
ObsData.Size	Estimated dataset size	OPT	

When multi-dimensional observational data is imaged, information may be lost. This is especially true for event and visibility data. Advanced client software that has the capability to deal directly with the event or visibility data corresponding to an image can use the ObsData reference for a Data element (if provided) to retrieve the observational data (the already imaged data array is available as well). Ideally the observational data will have been filtered to include only data samples directly contributing to the image area. The explicit connection of the ObsData element to a particular Data element makes such filtering possible, transparently to a client application.

ObsData.DataProductType and ***ObsData.DataProductSubtype*** are the dataproduct type and subtype as in the ObsCoreDM [ObsTAP-XX]. Typical values of *DataProductType* include "event" (for a reference to event data) and "visibility" (for a reference to interferometry data). *DataProductSubtype* specifies the type of data product as defined within the context of a specific archive or instrumental data collection. ***ObsData.CalibLevel*** defines the calibration level of the referenced data product as defined in ObsCore, e.g., 1 for instrumental data in a standard format, and 2 for calibrated, science ready data with the instrument signature removed.

The ObsData mechanism is intended mainly for images generated from event and visibility data, especially event data, for which analysis is typically done on the event domain, and the data volume tends to be manageable. ObsData can however be used for any type of observational data.

3.5 Additional Service-Defined Metadata

A given service may return additional query response metadata not defined by the ImageDM. This additional metadata may take the form of additional table columns, or additional RESOURCE elements in the query response VOTable.

Service-defined output metadata should use service-defined Utypes and UCDs as long as they do not clash - and can be easily distinguished - from mandatory and reserved ImageDM output columns. Extension metadata must conform to the rules for extension metadata as defined in the next section.

3.6 Metadata Extension Mechanism

The metadata extension mechanism allows a data provider to add additional custom metadata to the query response to describe collection-specific details of the data. . [*Add description of metadata extension mechanism, compatible with SpectralDM*].

4 Data Access Model

[*The following is copied from SIAV2 working draft and has not yet been fully integrated*. *The idea is to define the access model in terms of the ImageDM, i.e., the capabilities provided and how they map back to the Image model, but leaving how the access operation is formulated up to the specific, separately-defined data access protocol.*]

The *accessData* operation provides advanced capabilities for precise, client directed access to a specific image or image collection. Unlike *queryData*, *accessData* is not a query but rather a command to the service to generate a single image, and the output is not a table of candidate datasets but the actual requested image (or an error response if an error occurs). Use of *accessData* will generally require a prior call to *queryData* to get metadata describing the image or image collection to be accessed in order to plan subsequent access requests. *AccessData* is ideal for cases where an image with a specific orientation and scale is required, or for cases where the same image or image collection is to be repeatedly accessed, for example to generate multiple small image cutouts from an image, or to interactively view subsets of a large image cube.

4.1.1 Logical Access Model

The *accessData* operation is used to generate an image upon demand as directed by the client application. Upon successful execution the output is an image the parameters of which are what was specified by the client. The input may be an archive image, some other form of archive dataset (e.g., radio visibility or event data from which an image is to be generated), or a uniform data collection consisting of multiple data products from which the service automatically selects data to generate the output image.

In producing an output image from the input dataset *accessData* defines a number of transformations which it can perform. All are optional; in the simplest case the input dataset is an archival image which is merely delivered unchanged as the output image with no transformations having been performed. Another common case is to apply only a single transformation such as an image section or a general WCS-based projection. In the most complex case more than one transformation may be applied in sequence.

Starting from the input dataset of whatever type, the following transformations are available to generate the output image:

- **Per-axis input filter**. The spatial, spectral, temporal or polarization axis (if any) can be filtered to select only the data of interest. Filters are defined as a range-list of acceptable ranges of values using the BAND, TIME, and POL parameters as specified later in this section, for the spectral, temporal, and polarization axes respectively. POS and SIZE are specified as for *queryData* except that the default coordinate frame matches that of the data being accessed (more on this below). Often the 1D BAND, TIME, and POL axes consist of a discrete set of samples in which case the filter merely selects the samples to be output, and the axis in question gets shorter (for example selecting a single band of a multiband image or a single polarization from a polarization cube). In the case of axis reduction where an axis is "scrunched", possibly collapsing the entire axis to a single pixel, the filter can also be used to exclude data from the computation. Data which is excluded by a filter is not used for any subsequent computations as the output image is computed.

- **WCS-based projection**. This step defines as output a pixellated image with the given image geometry (number of axes and length of each axis) and world coordinate system (WCS). Since the input dataset has a well-defined sampling and world coordinate system the operation is fully defined. If the input dataset is a pixellated image the image is reprojected as defined by the new WCS. If the input dataset is something more fundamental such as radio visibility or event data then the input data is sampled or imaged to produce the output image. Distortion, scale changes, rotation, cutting out, axis reduction, and dimensional reduction are all possible by correctly defining the output image geometry and WCS.
- **Image section.** The *image section* provides a way to select a subset of a pixellated image by the simple expedient of specifying the *pixel* coordinates in the input image of the subset of data to be extracted (in our case here pixel coordinates would be specified relative to the image resulting from the application of steps 1 and 2 above). Axis flipping, dimensional reduction, and *axis reduction* (scrunching of an axis, combining a block of pixels into one pixel) can also be specified using an image section. *Dimensional reduction*, reducing the dimensionality of the image, occurs if an axis is reduced to a single value. The image section can provide a convenient technique for cutting out sections of images for applications that find it more natural to work in pixel than world coordinates. For example the section "`[*,*,3]`" applied to a cube would produce a 2D X-Y image as output, extracting the image plane at Z=3. Dimensional reduction affects only the dimensionality of the image pixel matrix; the WCS retains its original dimensionality.
- **Function**. More complex transformations can be performed by applying an optional transformation function to an axis (typically the Z axis of a cube). For example the spectral index could be computed from a spectral data cube by computing the slope of the spectral distribution along the Z axis at each point [x,y,z] in the output image.
- These processing stages define a *logical* set of transformations which can optionally be applied, in the order specifed, to the input dataset to compute the output image. Defining a logical order for application of the transformations is necessary in order for the overall operation to be well defined, as the output of each stage of the transformation defines the input to the following stage.

In terms of implementation the service is free to perform the computation in any way it wants so long as the result agrees with what is defined by the logical sequence of transformations. It is possible for example, for each pixel in the final output image, to trace backwards through the sequence of logical transformations to determine the signal from the input dataset contributing to that pixel. Any actual computation which reproduces the overall transformation is permitted.

In practice it may be possible to apply all the transformations at once in a single computation, or the actual computation may include additional finer-grained processing steps specific to the particular type of data being accessed and the sofware available for processing. The *AccessData* model specifies the final output image to be generated, but it is up to the service to determine the best way to produce this image given the data being accessed and the software available. The actual processing performed may vary greatly depending upon what type of data is accessed. [*We need to add some use cases to illustrate in concrete terms how this works.*].

Since *accessData* tells the service what to do rather than asking it what it can do, it is easy for the client to pose an invalid request which cannot be evaluated. In the event of an error the service should simply return an error status to the client indicating the nature of the error which occurred.

5 Serializations

[*The following is copied from the ImageDM section of the Cube whitepaper and has not yet been fully integrated*.]

Utype strings defined by the data model specification uniquely identify the elements of the Image data model, as in other VO data models. Aliases may also be defined for particular serializations, e.g., eight character FITS keywords, mapped one-to-one to data model Utypes, are defined to serialize an Image instance in FITS. Utypes and their aliases merely identify the fields of a data model *instance*. The semantics, usage, and meaning *of the data model itself* are defined separately from an instance, e.g., in the data model specification or in a schema of some sort.

The exact same Image instance may be represented in any number of forms by this technique without any loss of information (excepting possibly instance extensions not part of the formal data model). Instances may be converted from one serialization to another without loss of information.

Standard or optional Image serializations include the following:

- **FITS**. The primary standard for efficient binary representation of astronomical image data including multidimensional data cubes. Individual images may be represented in a single FITS file. Multiple images, e.g., Image sub-arrays as defined above, may be represented either as multiple distinct FITS images, as FITS image extensions in a multi-extension FITS file, or as the rows of a binary table. FITS WCS supports cube data including spatial, spectral, and polarization axes; full support for time is just now being standardized. The TAB WCS coordinate type supports sparse data axes. Image compression is supported.
- **VOTable**. VOTable is primarily used to represent “dataless” instances of the Image model, e.g., in data discovery queries where a dataless Image instance is used to describe and point to a remote dataset. VOTable could also be used to serialize images that include a data element; while not as storage or access efficient as FITS this could be useful for small image use cases, e.g., embedded preview images.
- **HDF5**. HDF5 is essentially a generic hierarchical container for data, similar to a hierarchical file system but with richer metadata, allowing large logically related collections of data objects to be efficiently stored as a single file. An Image instance can be represented as a single object in HDF5, or as a set of related objects, e.g., if

the Image instance has multiple sub-arrays. Within astronomy, LOFAR is using HDF5 for image (and other data) storage but supports FITS, CASA image tables, and other data formats for data export as well.

- **JPEG**. Graphics formats like JPEG are obviously important for graphical applications and are widely supported by a wealth of generic software outside astronomy. JPEG (and various other graphics formats) have the capability to embed arbitrary metadata directly in the image instance, hence this can be considered a form of Image serialization, although it is limited to 2-D images used for graphical use cases such as visualization.
- **Binary**. A special case of an Image serialization is the data segment of the Image instance with no associated header metadata, except possibly metadata defining the format (shape, depth, ordering, etc.) of the data array. This would be useful in applications where the Image instance metadata is known by other means. For example in a SIAV2 *accessData* operation, the client fully specifies the image data to be returned and there may be little need to return header metadata that would be redundant and probably ignored. This image format could improve performance in applications such as real time visualization and analysis.

In addition the following environment-specific formats are of interest:

- **CASA Image Table**. CASA (the radio data processing package used by ALMA and other projects) defines an *image table* format, in addition to FITS that is also supported. The image table format provides some flexibility in how the data element is organized. Unlike FITS that has a fixed, FORTRAN-array like ordering of image pixels or voxels, the CASA image table format supports additional options for ordering pixels, such as a blocked ordering which provides uniform time to access for any image axis.
- **Starlink NDF.** [*Add description here*.]

Other serializations may be defined, for example to support additional environments or tools. This list is intended only to describe some of the major image serializations, and the range of such serializations possible to support a wide range of applications.

Appendix A: Data Model Summary

Appendix B: Data Model Serializations

References

[1] **Plante R et al** (2007) **IVOA identifiers** http://www,ivoa.net/Documents/latest/IDs.html

[2] **Preite Martínez, A** (2007) **The UCD1+ Controlled Vocabulary** , http://www.ivoa.net/Documents/latest/UCDlist.html

[3] **Louys M. et al** (2011) **Observation Data Model Core Components and its Implementation in the Table Access Protocol v1.0,** http://www.ivoa.net/Documents/ObsCore/

[4] **Tody D. et al** (2011) **Simple Spectral Access Protocol (SSAP) v1.1**, http://www.ivoa.net/Documents/SSA/20110417/PR-SSA-1.1-20110417.pdf

[5] **McDowell J. et al** (2011) **IVOA Spectral Data Model v1.1,** http://www.ivoa.net/Documents/SpectrumDM/index.html