International

Virtual

Observatory

Alliance

IVOA Image Data Model

Version 1.0

IVOA Working Draft 2013 05 05

This version:
WD-ImageDM-1.0-20130505
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

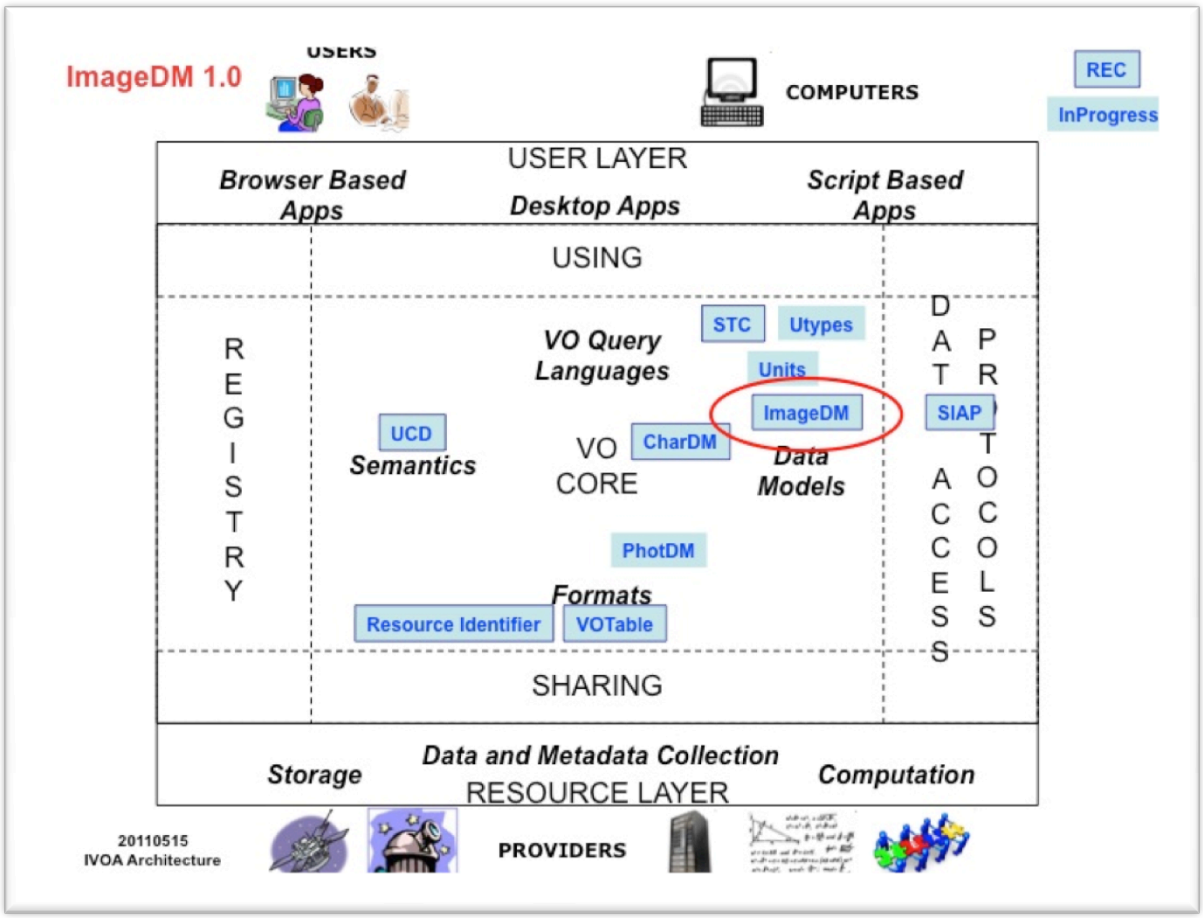
The Image Data Model (**ImageDM**) describes datasets characterized by an N-dimensional, regularly sampled numeric data array with associated metadata describing an overall "*image*" dataset. Image datasets with dimension greater than 2 are often referred to as "*cube*" or "image cube" datasets; in general either term may be used to refer to any n-dimensional image dataset. An image may have an associated *world coordinate system* (WCS) associating a physical scale with each measurement axis. The standard axes types 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, n-D 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 is emphasized, generalized *hypercube* or *n-cube* data including event and visibility data is also partially addressed.

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), including support 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, CASA image tables, and JPEG2000 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:



Status of This Document

The first release of this document was 2013 May 05.

This document has been produced by the IVOA Data Model Working Group.

It has been reviewed by IVOA Members and other interested parties, and has been endorsed by the IVOA Executive Committee as an IVOA Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. IVOA's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability inside the Astronomical Community.

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, probably to the introduction of the flexible image transport format (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.

A key aspect of the astronomical image is the 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 storage optimization and flexibility. This is especially important for large images or image cubes, which in the present era may be Gigabytes or Terabytes in size for a single dataset. Storing the data as a multidimensional numeric array allows generic software tools to be used for computation, e.g., the array processing capabilities of various scientific languages, or associated tools such as *NumPy* in Python. The N-d array (including cube data via various techniques) is relatively easy to render graphically as a conventional 2-D graphical image.

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 for more precise analysis (this is the approach adopted for the ImageDM, e.g., for event and visibility data).

Another 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 Motivation for a VO Image Data Model

Given that FITS is so widely used within astronomy, 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 N-d 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.

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 (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.

In the VO image or cube datasets may be very large, may be computed on the fly to match what the client requested (an example of *virtual data*), and may be used for any purpose, such usage being unknown to the data service. When scaling up, operations may need to be automated, requiring sufficiently detailed metadata to permit automated data selection or generation. To satisfy these diverse use cases the metadata returned in a discovery query needs to be fairly detailed, describing in some detail the characteristics of each candidate physical or virtual dataset.

The VO uses formal data models such as the ImageDM to describe the characteristics of physical or virtual dataset instances. Hence in the case of an image-specific discovery query, the metadata returned to describe each candidate image dataset *is an instance of the ImageDM*, minus the image data array of course. An actual image dataset ideally includes the same standard image metadata, plus possibly some additional dataset-specific custom metadata, plus the data segment for the image; the actual metadata returned will in general depend upon the application requirements and the data format used.

A VO data discovery query describes candidate datasets (N-d images in this case) in VOTable XML, whereas actual image datasets are commonly formatted as FITS images, both containing the same standard metadata describing the object. Hence we need to define the ImageDM as an abstract data model, independently of serialization, to satisfy our most basic query/access use case. More generally, 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 is considerably richer than what is defined in the FITS standards (although FITS is often extended via nonstandard 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 an N-d image (i.e., ImageDM) inherits from the more generic VO data models such as

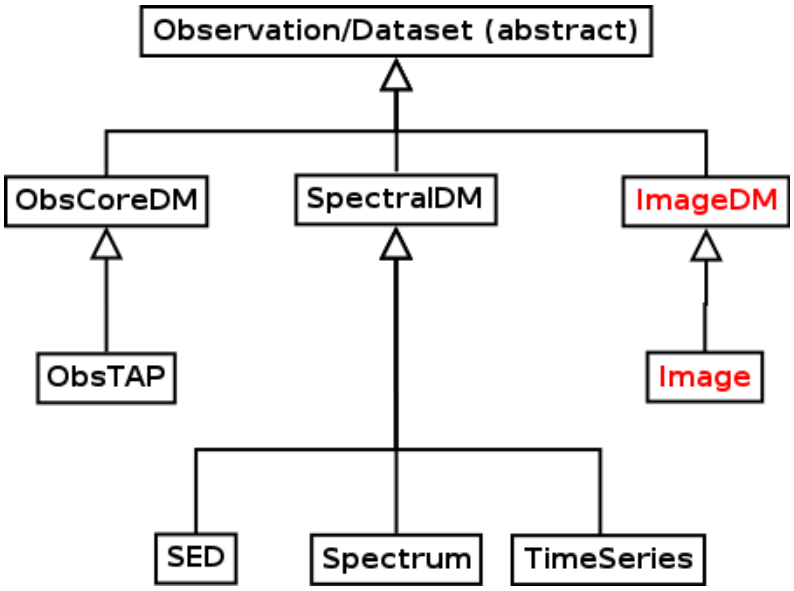
Observation and Characterisation. The data model for a specific class of data such as Image also needs to be extended to model the unique characteristics of the new class of data. In the case of ImageDM, the WCS submodel is a primary example of such an extension required for N-d image data. A convention 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.

2 Image Data Model

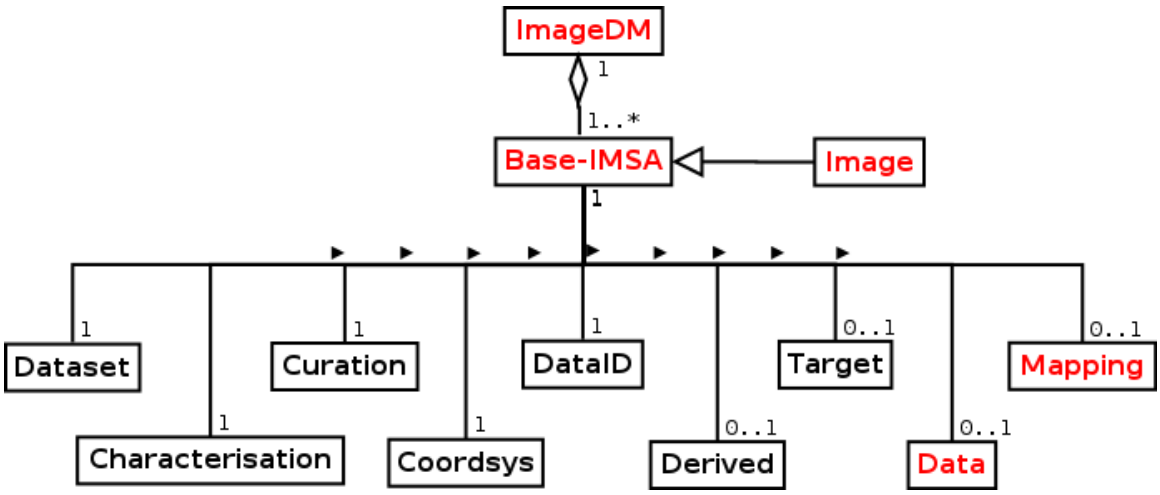
[*The following is copied from the ImageDM section of the Cube whitepaper and has not yet been fully integrated*.]

The *Image* data model provides the basis for discovery and access to astronomical "image" data in the VO.



Here "image" refers (in the simplest cases) to a multidimensional, regularly sampled numerical array with associated metadata describing the dataset instance. Unless dimensionality is otherwise indicated, the terms *image* and *cube* are interchangeable and both refer to N-dimensional image data. Image is a specialized

case of general *hypercube* data where the data samples are represented as a uniform multidimensional array of numerical values, allowing efficient computation and representation. A hypercube of dimension N is known as an *n-cube*. It follows that an N-dimensional image is a special case of an n-cube where the data samples are represented as a uniform N-dimensional numeric array. 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.



Typical instances of astronomical image data include a 2-dimensional FITS image and a 3 or 4-dimensional FITS image cube, or comparable image datasets in other formats. Although a spectrum may be represented as a 1-dimensional image, VO defines a more general *Spectrum* data model for spectra, and likewise for time series data. Visibility and event data may be considered a form of hypercube data in the most general sense; while the Image model does not directly address such data, both visibility and event data can be viewed as an image, and this is often done for reasons of efficiency and to allow generic image visualization and analysis software to be used.

As with most VO data models, the Image data model is defined as an abstract model independently of how it is realized, or serialized in some particular data stream or file format. Image inherits generic dataset metadata from common VO data models such as *Observation*, *Characterization*, and Space Time Coordinates (STC). Other VO data models such as *Spectrum* and its underlying Spectral data model (SDM) are largely the same as Image, all being derived from the same root generic dataset models. Image also inherits from and is semantically compatible with the FITS image and FITS world coordinate system (WCS) models. Unlike traditional FITS however, Image may be serialized in a variety of data formats including but not limited to the FITS serialization (this approach could provide a way forward to modernize FITS and free it from its overly restrictive legacy serialization).

2.1 Abstract Model

The abstract Image data model provides a unified description of simple ND-image instances, sparse images/cubes, and indirectly, generalized hypercube data such as visibility and event data.

The root abstract data model is a collection of sub-arrays/images that all fit into a single super-array. The metadata (including WCS) for the super-array must be provided to describe the overall Image dataset. WCS metadata for each sub-array needs to be provided with that sub-array. The sub-arrays in the collection need not be the same size.

- With this model, a standard single ND-image is a special case — the collection of sub-arrays may consist of just one sub-array that is congruent with the super-array, for example a single 2D or ND image.
- In the sparse image case, regions of the multidimensional image space have no data. An individual sub-array may be sparse, the area of the super-array may be only partially covered by the provided sub-arrays, or both cases may occur simultaneously.
- Support for generalized hypercube data (typically visibility or event data) is provided only indirectly. By default, Image provides an image view of such data. A reference (possibly in the form of a VO *datalink*) to the underlying more fundamental data may optionally be provided, allowing the client to retrieve the underlying data and work with it directly. Ideally the referenced hypercube data should be filtered (in all dimensions) to return only data within the region covered by the image view.

An instance of the Image data model may contain the image data array or arrays, or may contain only metadata plus a reference to externally stored data. The referenced external data may be a static data object or may be ***virtual data*** that is computed upon demand when accessed. This is used for example in data queries where the Image instance refers to an available remote dataset, or in pass-through of hypercube data where the Image instance may contain embedded information for how to filter and generate the remote event or visibility data corresponding to the Image instance.

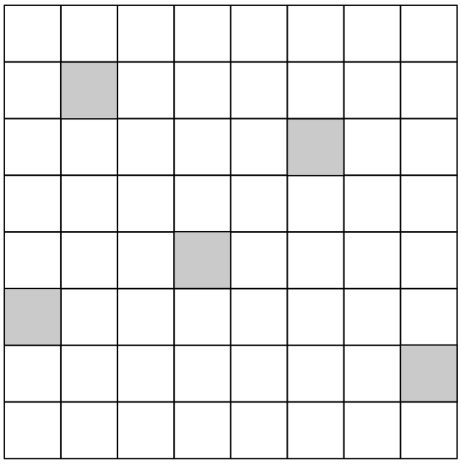
2.1.1 Hypercube Data

Image allows the format of any referenced event or visibility data to be queried, but the actual format used is determined and defined externally (e.g., by the data provider or by some external standard) rather than by the Image data model. For this to be useful the client must be able to deal with the provided data, however it may be possible to perform advanced analysis by working directly with the more fundamental hypercube data. An X-ray analysis tool for example, might work directly with the event data for an observation, performing multiwavelength analysis combining the event data with image data from other sources. A generic image analysis tool could perform a similar analysis using the image view of the same image dataset.

2.1.2 Sparse Data

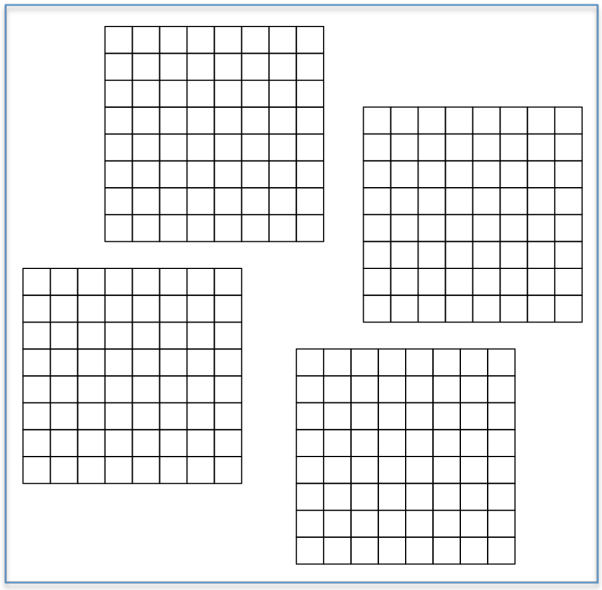
Given a single sparse sub-array (ND-image), each sparse image axis may be indexed by a table giving the coordinates of each provided sample (FITS WCS for example defines a TAB coordinate type for this purpose). Any image axis may be sparse; arbitrary or irregular sample spacing may also be represented using this technique. For example, an irregularly sampled spectral, time, or polarization axis may specify the specific coordinates at which each data sample is provided.

More subtly, the 2D spatial plane may contain data only at specific locations within the 2D plane. If the coordinates of a sample point are specified in floating point they may be randomly located at WCS-defined coordinates within the covered region, for example the sampled points may correspond to the observed sources within a field. If the coordinates of each sample point are expressed as integer values then the spatial plane is regularly sampled (pixelated) but sparse, with data provided only for data samples that are indexed. For example, if we have a spectral image cube that is sparse in the spatial plane with 4K resolution elements along the spectral axis, the data array would consist of N vectors each of length 4K, where N is the number of sampled points in the spatial plane. The associated index table might define the RA and DEC of each sampled spatial point in the cube, or the *(i,j)* pixel coordinates of each sampled point in the case of a sparse pixel array.



Example of a **sparse image** (image or image cube which is sparse on the two coupled spatial axes). Data was obtained only for the points shown as gray in the figure. Rather than store the entire array, only data for the five sampled regions is stored. The coordinates of each sampled region are stored in a table included in the WCS for the image/ cube. In this example the sparse cube would be represented in 5/64 of the space that would be required to store the fully sampled cube.

Sparse data represented by multiple sub-arrays is more verbose, but simpler in some respects. In this case the overall dataset consists of a super-array containing overall dataset metadata and optional image data for the super-array region, plus some number of sub-arrays providing image data for the covered regions. The sub-arrays may or may not share the same sampling and coordinate frame / projection. An example might be a spatial field where data is available only for several sub-fields, or a spectral data cube where data is available only for several widely spaced spectral bands. The specific serialization used defines how to aggregate multiple



Example of a **sparse image** (image or image cube which is sparse on the two coupled spatial axes), that is composed of several sub-arrays. The outer box defines the area of the super-array, or overall Image dataset. The four sub-arrays are individual smaller images for which data was obtained. This example illustrates the use of multiple sub-arrays to cover a larger spatial region, however the same technique may be used for other axes such as the spectral, time, and polarization axes of a general cube.

sub-arrays within a single dataset.

3 Data Model Classes

[*Define each class element of the ImageDM and relate to more fundamental data models such as Observation and Characterisation. The following is copied from the relevant section of the SIAV2 specification and has not yet been fully integrated.*]

3.1 Types of Metadata

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

Service Metadata	
Query	Describes the query itself
Association	Logical associations
Access	Dataset access-related metadata
Data Model Metadata	
Dataset	General dataset metadata
DataID	Dataset identification (creation)
Provenance	Instrumentalm or software Provenance
Curation	Publisher metadata
Target	Observed target, if any
CoordSys	Coordinate system frames
Char	Dataset characterization
Mapping	Dataset Axes Mapping or WCS
Characterization Metadata	
Char/FluxAxis	Observable, normally a flux measurement
Char/SpectralAxis	Spectral measurement axis, e.g., wavelength
Char/TimeAxis	Temporal measurement axis
Char/SpatialAxis	Spatial measurement axis
Char/Polarization	Polarization Axis
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
Mapping metadata	
Image matrix mapping	
WCS Mapping	

Service metadata is specific to the functioning of the service itself, for example to step through large queries or retrieve selected datasets. **Data model metadata** describes each dataset. **Characterization metadata** physically characterizes the

dataset in terms of the spatial, spectral, and temporal measurement axes and the observable. Characterization is part of the data model but is broken out separately in the table above to show the major elements of the characterization model. Most image metadata is generic dataset metadata that can be used to describe any type of data.

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

3.2 Query Metadata

Query metadata describes the query itself.

UTYPE	Description	Req
Query.Score	Degree of match to query params	REC

3.2.1 Query.Score

A record with a higher score more closely matches the query parameters. The score is expressed as a floating point number with an arbitrary scale (different queries may return results with different scale factors and cannot be inter-compared). If scoring is used, the query response table should be returned sorted in order of decreasing values of score, with the top-scoring items at the top of the list. The details of the heuristic used to compute the score are left to the service. See the discussion of the `TOP` parameter in section …..

3.3 Association Metadata

Association metadata is used to describe logical associations relating datasets described in the query response, as described in section … . Logical associations between query response records may refer to the data access operation itself, e.g., where the same data object is available in multiple output formats, or to logical associations relating the physical data, e.g., where multiple primary datasets are part of the same observation. The same dataset may belong to multiple associations.

UTYPE	Description	Req
Association.Type	Type of association	OPT
Association.ID	Unique ID identifying the association instance	OPT
Association.Key	Unique key different for each element of association	OPT

Each such association is described by a separate instance of the Association model, with a defined Association Type, ID, and Key. In many cases the Association Type and Key can be represented as fixed PARAMs, leaving only Association.ID to be represented as a FIELD in each table row.

In general, specification of the allowable Association types is beyond the scope of this specification. The semantic details of Associations are intended to be defined either at a lower level, for a specific data collection or service, or at a higher level, e.g., to describe complex data associations. An exception is the MultiFormat association described in the next section.

3.3.1 MultiFormat Association

A pre-defined case is the *MultiFormat* association, where several records refer to the same dataset which is available in several different output data formats. In this case Association.Type should be set to "MultiFormat", Association.ID can be anything, and Association.Key should be set to "@Access.Format" to indicate that the key which differentiates the elements of the association is the value of the Access.Format field of the record. If several query response records are of this type the association **should** be specified to indicate the association. In all other cases (currently undefined by the protocol) the association **may** be specified.

3.3.2 Association.Type

A service-defined type used to indicate what type of association is being referred to. The value should be unique within the scope of the query response. There can be many types of logical associations. Associations provide a means of describing complex data aggregations relating multiple datasets. Examples of possible associations might be a multi-CCD detector observation consisting of as many images as CCD, each of which appears in the query response as an individual image, or a group of query response records which all refer to the same dataset but differ only in the output format.

Since the association type may be shared by many table records, it may be best specified as a PARAM in the output VOTable, using an ID-REF to link it to the association it refers to. An association type **should** be provided for each association in the table.

3.3.3 Association.ID

The association ID is a string, unique within the scope of a given VOTable, identifying one instance of a given association. All members of the association instance share the same Association.ID. The association ID **must** be provided for any association. The content of the string is up to the service. Multiple association IDs may be provided for a single record if a record belongs to more than one association. Note that Association.ID is unrelated to the VOTable ID, which is used to uniquely identify the elements of a VOTable.

Extension metadata may optionally be provided to describe an association in more detail. Extension metadata appears in the output VOTable as optional additional RESOURCE elements (see section ….). The ID-REF mechanism may be used to link such an extension record to the association in the main table. The contents of an association metadata extension record are externally defined by the application using the data model.

3.3.4 Association.Key

The association key **should** be provided to identify what is "different" for each member of an association. The value is a string and may be either an arbitrary value defined by the association, or a reference to one or more table fields which form the association key. If a table field is referenced the '`@`' character should be prefixed to the VOTable ID of the referenced FIELD to indicate the indirection (e.g., "`@Format`"), otherwise the literal string is used as the key. A key may contain multiple elements delimited by commas.

3.4 Access Metadata

Access metadata is required to tell a client how to access the datasets described in the SSA query response.

UTYPE	Description	Req
Access.Reference	URI (URL) or template used to access the dataset	MAN
Access.Format	MIME type of dataset	MAN
Access.Size	Estimated (not actual) dataset size	REC
Access.Parameters.*		OPT

3.4.1 Access Reference

The simplest case of access reference is a URI (typically a URL) which can be used to synchronously retrieve the specific dataset described in a row of the query table response. If the dataset pointed to by the access reference does not exist at query time, it will be computed on the fly when accessed.

Access Reference can also be an URI template if some of the PARAMETERS can be filled interactively by the client during the AccessData phase. Access.Parameters will help to do this.

SIAV2 supports data staging and asynchronous data access. Support for these functionalities is described below and is useful to support generation of simulated or synthetic data, as well as very large images retrieval.

When the access reference is a URL, it is convenient to be able to input the access reference directly in a Web browser or other standard Web tool to access the referenced dataset. For this reason the access reference string should be URL-encoded if it contains any reserved URL metacharacters (the "#" character used in dataset identifiers is particularly nasty). See also section… . The CDATA construct used in earlier data access interfaces (SIAP V1.0) does not serve the same purpose and should not be used; use URL encoding instead.

3.4.2 Output Format

The file format of a candidate dataset is specified by its MIME type. Both uncompressed and compressed data can be indicated in this fashion.

The file format says nothing about the data model used by whatever data object is stored in the file; this is specified by the Dataset.DataModel attribute discussed in section .....

A single data object may be available in multiple file formats. In such a case an association **should** be defined to indicate that the entries all refer to the same data object.

3.4.3 Dataset Size Estimate

The approximate estimated size of the dataset, specified in kilobytes, **should** be given to help the client estimate download times and storage requirements when generating execution plans. Only an approximate, order of magnitude value is required (a value rounded up to the nearest hundred KB would be sufficient). In the VO dataset sizes can vary by many orders of magnitude hence it is important to know this information to optimize execution plans before attempting to download data or request computation. It is preferable to return an order of magnitude estimate of the dataset size, than no value at all. A precise value is *not* required.

3.4.4 Access Parameters:

[*This should probably be handled by getCapabilties instead but is left in here for the present to record the type of functionality which we need to describe.]*

These parameters allow to describe detailed and specific access modes to the data:

- Access.param.Interactive gives the list of interactive parameters providing optionnaly possible ranges of values (eg POS, SIZE, COMPRESS, etc…)
- In order to give information to the client of where to find appropriate information in the retrieved file a couple of UTypes have been defined. Such specification is made necessary because sometimes the actual science data is only a subpart of the retrieved file :
- Acces.param.extnum and Access.param.extname give the Extension number and Extension name in FITS (or VOTABLE)
- Access.param.Cutout gives the cutout limit (à la “IRAF”) in Fits Array,
- Access.param.Field and acces.param.row give the name/number of the Field/row in FITS table or VOTABLE….)

3.5 Data Model Metadata

The following metadata components are in common with other VO data models such as ObsCore and the Spectral data model.

3.5.1 General Dataset Metadata

General dataset metadata describes the overall dataset.

UTYPE	*Description*	*Req*	*Default*

Dataset.DataModel	Datamodel name and version	MAN	Obs-1.0
Dataset.Type	Type of dataset	MAN	Image, cube
Dataset.Length	Number of pixels in image/cube	MAN	
Dataset.Deleted	Set to deletion time, if dataset is deleted	OPT	

Dataset.DataModel is a string identifying the data model type and version used in the described dataset. For Image-compliant data this should be a value such as "Obs-1.0", as specified in the Obs data model document for the version of the data model being used. For pass-through of native project data some other value should be used which identifies the specific project data model used, e.g., "HST-STIS-1.0".

For the ImageDM, Dataset.Type is either "Image", or “Cube”. Dataset.Length is mandatory and specifies the dimensionless "length" of the image, i.e., the total number of pixels or samples in the full image. Dataset.Deleted is used with the MTIME query parameter to inform the client that a previously existing dataset has been deleted; if a service supports MTIME it should also support Dataset.Deleted. The value is the ISO 8601 date (as in MTIME) at which the dataset was deleted, or null for a normal non-deleted dataset. Dataset.Deleted should be returned in a query only if MTIME is used in the query, and the deletion date matches the interval of time specified by MTIME. Otherwise deleted datasets should never be visible in a query. A service may permanently delete dataset deletion history after a period of time (currently unspecified) long enough to permit clients to discover deleted datasets.

3.5.2 Dataset Identification and Provenance Metadata

Dataset identification metadata is used to describe the fundamental identify of a dataset, including where it came from and how it was created.

UTYPE	*Description*	*Req*	*Default*
DataID.Title	Dataset title	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	REC	
DataID.Date	Data processing/creation date	OPT	
DataID.Version	Version of creator-produced dataset	OPT	
DataID.CreationType	Dataset creation type	REC	archival
Provenance.ObsConfig.Instrument	Instrument name	OPT	
DataID.ObsConfig.Bandpass	Bandpass name, e.g., filter	OPT	
DataID.ObsConfig.DataSource	Original source of data	REC	survey

Dataset.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, filter,

target name, etc., is typically included in a condensed form. The exact contents of Dataset.Title are up to the data provider. Dataset.Creator identifies the entity which created the dataset, and should be a short string consistent with the RSM specification, e.g., "SDSS". Dataset.Collection is the registered IVOA identifer of the data collection to which the dataset belongs, e.g., "ivo://sdss/dr5/spec".

The CreatorDID is the 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 (see …. for further details). 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, as defined in section … .

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

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 I, J, K, Q, HI, and so forth. Values specified with Provenance.ObsConfig.Bandpass may be used as input to the BAND parameter (…..) to refine a query (if this feature is supported by the service).

Provenance.DataSource describes the original source of the data.

3.5.3 Curation Metadata

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

UTYPE	*Description*	*Req*	*Default*
Curation.Publisher	Publisher	MAN	
Curation.Reference	URL or Bibcode for documentation	REC	
Curation.PublisherDID	Publisher's ID for the dataset	REC	
Curation.Date	Date curated dataset last modified	OPT	
Curation.Version	Version of curated dataset	OPT	
Curation.Rights	Restrictions: public, proprietary, etc	OPT	public

Curation.Publisher is a short string identifying the publisher of the data, e.g., a data archive or data center, or an indexing service such as the ADS. Curation.PublisherDID is the IVOA dataset identifier (URI) assigned by the publisher to identify the dataset within its holdings. Curation.Reference is a forward link to

publications which reference the dataset; multiple instances are permitted. Curation.Date 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.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.

Note:
If the same dataset is replicated at several locations with multiple publishers, it is possible to set up an association group to indicate this fact.

3.5.4 Astronomical Target Metadata

Target metadata describes the astronomical target observed, if any.

UTYPE	*Description*	*Req*	*Default*
Target.Name	Target name	OPT	
Target.Class	Target or object class	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.Class is the object class if known, e.g., Star, Galaxy, AGN, QSO, and so forth (see section). Target.Redshift, Target.VarAmpl, are as defined in the data model. Either standard target values, or derived quantities, may be used in the query response.

3.5.5 Coordinate System Metadata

Coordinate system metadata describes the coordinate system reference frames used in the ImageDM instance.

UTYPE	*Description*	*Req*	*Default*
CoordSys.SpaceFrame.Name	Spatial coordinate frame	REC	ICRS
CoordSys.SpaceFrame.Equinox	Equinox	OPT	2000.0
CoordSys.TimeFrame.Name	Timescale	OPT	TT
CoordSys.TimeFrame.Zero	Zero point of timescale in MJD	OPT	0.0

These reference frames apply to all spatial (sky), spectral, and time coordinates used in the ImageDM instance (including Characterization) unless otherwise specified. 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.

3.5.6 Dataset Characterization Axis Metadata

The Characterization axis metadata specifies the type of physical quantity on each physical measurement axis as well as the observable.

UTYPE	*Description*	*Req*	*Default*
Char/FluxAxis.Ucd	ucd for flux	REC	
Char/SpectralAxis.Ucd	ucd for spectral coord	REC	
Char/TimeAxis.Ucd	ucd for time coord	REC	
Char/SpatialAxis.Ucd	ucd for time coord	REC	
Char/PolarizationAxis.Ucd	Ucd for pol axis	REC	

Values are specified as UCDs, as defined in the data model. For example, to specify that the flux axis is flux density per unit wavelength, the value "phot.fluDens;em.wl" would be given.

3.5.7 Characterization Coverage Metadata

The Coverage component of the Characterization data model (Char) describes the coverage of the dataset in each of the four primary measurement axes.

UTYPE	*Description*	
Char/SpatialAxis.Coverage.Location.coord	Observed position, e.g., RA DEC	MAN
Char/SpatialAxis.Coverage.Bounds.Extent	angular area, sq deg	MAN
Char/SpatialAxis.Coverage.Bounds.limits.LoLimit2Vec		
Char/SpatialAxis.Coverage.Bounds.limits.HiLimit2Vec		
Char/SpatialAxis.Coverage.Support.AreaType		
Char/SpatialAxis.Coverage.Support.Area	Accurate Field of View	OPT
Char/TimeAxis.Coverage.Location.coord	Midpoint of exposure (MJD)	MAN
Char/TimeAxis.Coverage.Bounds.Extent	Total elapsed exposure time	REC
Char/TimeAxis.Coverage.Bounds.limits.LoLimit	Start time	OPT
Char/TimeAxis.Coverage.Bounds.limits.HiLimit	Stop time	OPT
Char/TimeAxis.Coverage.Support.Extent	Effective exposure time	OPT
Char/SpectralAxis.Coverage.Location.coord	Midpoint of Spectral coord range	MAN
Char/SpectralAxis.Coverage.Bounds.Extent	Width of spectrum in meters	MAN
Char/SpectralAxis.Coverage.Bounds.limits.LoLimit	Start in spectral coordinate	REC
Char/SpectralAxis.Coverage.Bounds.limits.HiLimit	Stop in spectral coordinate	REC
Char/PolarizationAxis.enumeration		

Within Char, Coverage specifies the *location* (central or characteristic value), *bounds* (measurement limits), *support* (region covered within the bounds), for each measurement axis. The coordinate system reference frames specified in Coordsys apply here. Spatial coordinates are specified in units of decimal degrees, spectral

coordinates in units of meters, and time coordinates in units of days. The Polarization axis is peculiar in this that it gives the list of available polarization parameters for the polarization system given by the UCD (eg Q, U, V parameters for Stokes system….)

3.5.8 Characterization Resolution and Sampling Metadata

The `Resolution` component of Characterization specifies the sampling and resolution estimates for the dataset.

UTYPE	*Description*	*Req*	*Default*
Char/SpectralAxis.Resolution	Spectral res. FWHM	REC	*BinSize*
Char/TimeAxis.Resolution	Temporal res. FWHM	OPT	*BinSize*
Char/SpatialAxis.Resolution	Spatial resolution of data	REC	
Char/SpectralAxis.SamplingPrecision.RefVal	Wavelength bin size	OPT	
Char/TimeAxis.SamplingPrecision.RefVal	Time bin size	OPT	
Char/SpectralAxis.SamplingPrecision.FillFactor	Sampling filling factor	OPT	1.0
Char/SpatialAxis.SamplingPrecision.FillFactor	Sampling filling factor	OPT	1.0
Char/TimeAxis.SamplingPrecision.FillFactor	Sampling filling factor	OPT	1.0

The spatial and spectral resolution **should** be specified. Note that, for consistency within Char, the spectral resolution is specified here in spectral coordinate units (FWHM in meters), unlike the SPECRP query parameter, which is specified as $\lambda/d\lambda$.

3.5.9 Characterization Accuracy and Error Metadata

The `Accuracy` component of Characterization specifies the sampling, resolution, and error estimates for the dataset.

UTYPE	*Description*	*Req*	*Default*
Char/FluxAxis.Accuracy.StatError	Statistical error	OPT	
Char/FluxAxis.Accuracy.SysError	Systematic error	OPT	
Char/FluxAxis. CalibrationStatus	Type of flux calibration	REC	calibrated
Char/SpectralAxis.Accuracy.StatError	Spectral coord meas. error	OPT	
Char/SpectralAxis.Accuracy.SysError	Spectral coord meas. error	OPT	
Char/SpectralAxis. CalibrationStatus	Type of coord calibration	REC	calibrated
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	calibrated
Char/SpatialAxis.Accuracy.StatError	Astrometric statistical error	REC	
Char.SpatialAxis.Accuracy.SysError	Systematic error	OPT	
Char.SpatialAxis. CalibrationStatus	Type of coord calibration	REC	calibrated

Both overall statistical and systematic error estimates may be specified. The calibration status of all three primary measurement axes as well as the observable **should** be given, otherwise "calibrated" is assumed.

3.6 Mapping Metadata

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

UTYPE	*Description*	*Req*	*Default*
Image Matrix Transform			
Mapping.NAxes	Number of image axes		
Mapping.NAxis[]	Length of each axis in pixels		
Mapping.CoordRefPixel[]	Reference pixel		
Mapping.CoordRefValue[]	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		
Mapping.WCSAxes	Number of WCS axes		
World Coord Transform			
Mapping.SpatialAxis.CoordType	Coordinate type as in FITS		
Mapping.SpatialAxis.Projection	Celestial projection		
Mapping.SpatialAxis.CoordFrame	Spatial coordinate frame		
Mapping.SpatialAxis.CoordEquinox	Coordinate equinox (if used)		
Mapping.SpatialAxis.CoordUnit	Unit for coordinate value		
Mapping.SpatialAxis.CoordName	Axis name (optional)		
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.CoordValue[]	Spectral value/band at pixel index		
Mapping.TimeAxis.CoordType	Time scale (UTC, TT, TAI, ...)		
Mapping.TimeAxis.CoordUnit	Time unit		
Mapping.TimeAxis.CoordName	Time axis name (optional)		
Mapping.TimeAxis.CoordValue[]	Time value at pixel index		
Mapping.TimeAxis.RefPosition	TOPOCENT, BARYCENT, ...		
Mapping.PolAxis.CoordType	Polarization system (Stokes etc.)		
Mapping.PolAxis.CoordName	Polarization axis name (optional)		
Mapping.PolAxis.CoordValue[]	Polarization type at pixel index		

In the above table, UTYPEs that have "[]" appended are vector-valued (the value is a string consisting of a sequence of numbers or string tokens delimited by spaces). The "[]" is not part of the actual UTYPE.

The Mapping model used in the ImageDM is essentially the same as FITS WCS although it is represented slightly differently and has been somewhat simplified. The biggest deviation is in the representation of polarization that is represented as a simple lookup table assigning a polarization type to each pixel index on the polarization axis (e.g, "I", "Q", "U", "V" for full Stokes).

A detailed description of the FITS WCS model is beyond the scope of the current document but can be found in FITS WCS papers 1-5. In summary 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.

In our representation the *AxisMap* is then used to map the axes of the intermediate world coordinates 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 coordinte values. 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 wherein each pixel index on the axis is directly assigned a world coordinate.

[*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.*]

3.7 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.

3.8 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.

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.
- **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.
- **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.

Other serializations may be defined, e.g., the **Starlink NDF** format is similar in capabilities to the above. 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 a**l (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