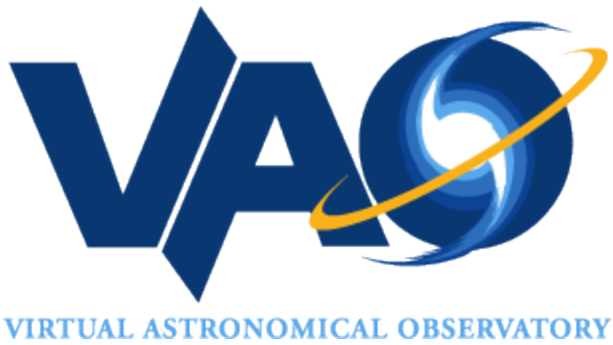

Simple Image Access Protocol

Version 2.0

VAO Prototype Baseline 2013 August 30

This version:
ThisVersion-YYYYMMDD
Latest version:
http://www.ivoa.net/Documents/latest/latest-version-name
Previous version(s):
http://wiki.ivoa.net/internal/IVOA/SiaInterface/WD-SIAP-2.0-20091104.pdf

Author(s):
D. Tody, F. Bonnarel R. Plante
Author2

Editor(s):

Abstract

The Simple Image Access Protocol version 2.0 (**SIAP** or **SIA** depending upon the context) provides capabilities for the discovery, description, access, and retrieval of multi-dimensional *image* datasets, including 2-D images as well as cubes of three or

more dimensions. SIAP is based upon the Image Data Model (**ImageDM**), which describes datasets characterized by an N-dimensional, regularly sampled numeric data array, with associated metadata describing the overall multi-dimensional image dataset as well as a separable World Coordinate System (**WCS**) describing the physical measurement axes of the image. 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 unless the image dimensionality is otherwise specified.

SIAP version 2.0 provides a wide range of capabilities for image discovery and access. At the most basic level the protocol allows whole archival image datasets to be discovered and retrieved via a simple HTTP REST-based interface (hence the simple use case implied by the "simple" in the name SIAP). More complex optional advanced capabilities are defined for directly accessing remote image datasets without having to retrieve the entire dataset. The logical model for an image dataset as seen externally is independent of how the dataset is physically stored, allowing sophisticated back-end storage schemes or data formats to be used for large image datasets. Capabilities are provided to filter, subset, and transform image datasets, allowing arbitrarily large images or cubes to be remotely accessed. Access operations may be either synchronous or asynchronous and multiple concurrent access operations may execute simultaneously, providing scalable access to large datasets. Image datasets may be sparse on any combination of image axes. Image data may be returned in a variety of output formats, including custom formats, with FITS being the default standard image format.

Status of This Document

This document is intended to represent a close approximation to the SIAv2 Working Draft currently under development within the IVOA. Based on an intermediate version of the IVOA working draft developed within the IVOA Data Access Layer (DAL) working group, it was adapted specifically to guide development of a prototype SIAv2 service by the Virtual Astronomical Observatory (VAO) project. For the purposes of controlling scope and evaluating the results of the prototype, this document is versioned and change-controlled by the VAO project. This draft carries no endorsement by the DAL-WG nor the IVOA in general, and it should not be considered as part of the official line of versions maintained by the DAL-WG. The latest official working draft (internal or otherwise) can be found at http://wiki.ivoa.net/twiki/bin/view/IVOA/SiaInterface.

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

Acknowledgements

"Ack here, if any"

1 Introduction

The Simple Image Access Protocol (SIAP or SIA depending upon the context) provides capabilities to discover, describe (via standard metadata), retrieve, and access astronomical image datasets of any dimension. Typical image datasets include 2-D spatial images, spectral data cubes, and cube and hypercube data of higher dimensions as well as derived image data products. While the data model is general, typical image datasets are derived from observational data and have some combination of spatial, spectral (including velocity and redshift), time, and polarization axes. Data access capabilities range from simple file retrieval to dynamic generation of virtual data (virtual images) that filter, subset, or transform a remote image dataset.

1.1 Role Within the IVOA Architecture

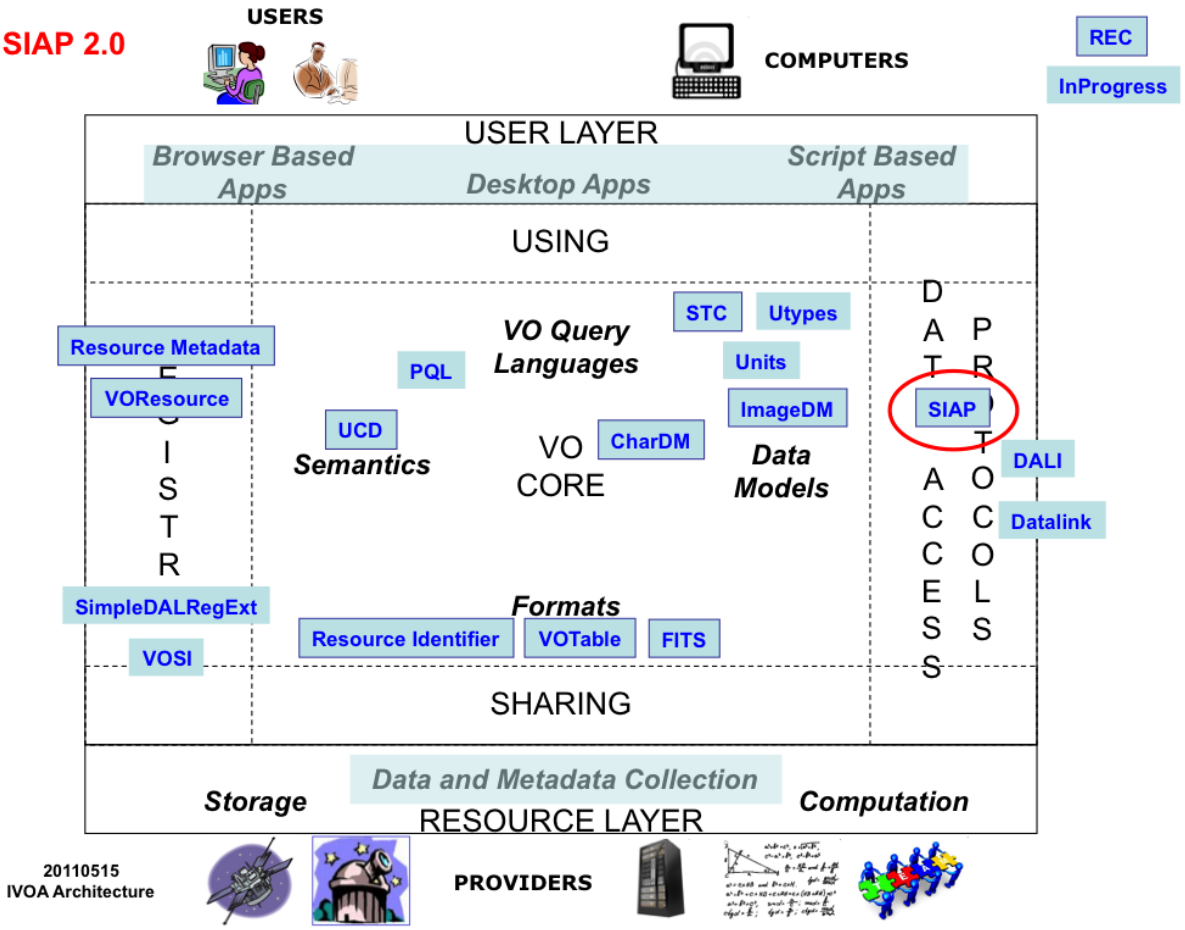

Figure 1. The role of the Simple Image Access Protocol (SIAP) in the IVOA architecture.

SIAP version 2.0 depends upon the IVOA Image data model (**ImageDM**, and indirectly FITS and **FITS WCS**) for a specification of the general N-dimensional astronomical image data model upon which the data access protocol is based. The Data Access Layer (DAL) Interface specification (**DALI**) specifies the interface elements common to SIAP and other IVOA DAL protocols. The DAL architecture identifies the Parameter Query Language (**PQL**) specification to define the common syntax of parameterized

interfaces in DAL services, however this specification is not yet sufficiently advanced for use by SIAP version 2.0, so the Simple Spectral Access (SSA) protocol is used instead. SSA V1.1 (Feb 2012) predates PQL and includes a section on basic service elements defining the standard "simple DAL" parameter syntax. The Datalink standard, still in the working draft stage, specifies how to link related resources such as documents, datasets, graphics products, or services to a given data product or observation. Datalinks may appear in a SIA query response, but are not required to use the interface. Other generic standards such as for dataset characterisation (CharDM), space-time coordinate metadata (STC), and VOTable are fundamental to all IVOA services. VOSI specifies the standard elements of VO services for things like service capabilities and monitoring service availability. The simple DAL registry extension (SimpleDALRegExt) is the basis for registering "simple DAL" services like SIAP in the VO resource registry.

Although SIA depends directly or indirectly upon all these IVOA standards, the current document summarizes the essential elements required to use the protocol. Further details are given in the related standards documents. References to these documents are indicated herein with a tag optionally followed by a section number, all enclosed in square brackets, e.g., "[DALI-4.2]" would refer to section 4.2 of the document tagged as "DALI" in the References section of this document. Section numbers are guaranteed to be accurate only for the specific version of the document referenced.

1.2 Image Data Model

The Image Data Model (ImageDM) is the basis for the image access protocol, and is a formalization of the familiar concept of a multi-dimensional astronomical "image". Full details on the image data model are presented in the *IVOA Image Data Model* specification [ImageDM], which we summarize briefly here. The VO Image data model is based upon and compatible with FITS, but defines additional VO-specific metadata and allows image datasets to be serialized in a variety of data formats.

The Image data model 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. The data samples of an image are referred to as *pixels* (picture elements), or more generally as *voxels* (volume elements).

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 is emphasized, limited support for more fundamental multi-dimensional observational data including event and visibility data is included via a pass-through mechanism referencing external observational data.

1.2.1 Data Model Architecture

The elements of the Image data model are illustrated in Figure 2. A **simple image** dataset has a single *Data* element containing the N-dimensional data array and associated WCS (*Mapping* element). Standard VO dataset metadata describes the overall image dataset. The Data element may be omitted if an Image instance merely describes an image dataset.

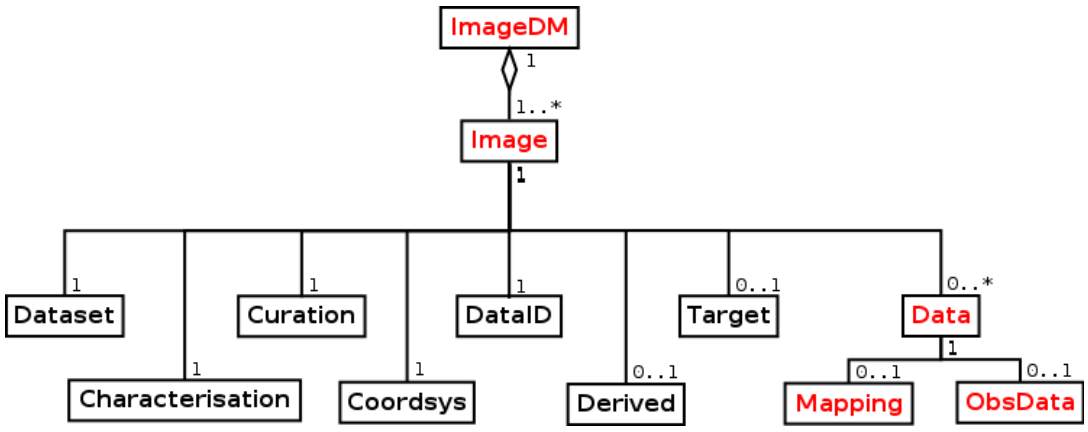

Figure 2. The Image data model.

Sparse Data. An image dataset may have multiple Data elements if the data is sparse along one or more image axes. Each such Data element has its own WCS and other data element-specific metadata. A single Data element may also be sparse, using the WCS associated with that Data element to assign coordinates to an explicit list of data samples.

Complex Data. Multiple image dataset instances (and possibly other types of datasets) may form an *association*, e.g. in the SIAP query response table, to model data objects too complex to be represented by a single image dataset.

Data Linking. An image dataset may have zero or more *data links* [11] pointing to auxiliary data products, services, or other resources associated with the specific image dataset.

1.2.2 Serializations

Like most VO data models, the ImageDM is defined as an abstract data model, independently of how it is realized or serialized. Any number of serializations (data formats) may be defined. An Image instance may be serialized in any such format, or may be converted between any two different formats, without loss of information, with the possible exception of format-specific extensions.

Two standard serializations are defined as part of the ImageDM:

- **VOTable**. The XML-based VOTable format [VOTable] is used primarily for data discovery queries and for metadata retrieval. In the case of a discovery query an Image instance (dataless) is serialized as a table row, with each row of the query response table describing a single image dataset. VOTable format may also be

used to retrieve the metadata for a single image dataset, e.g., to get full metadata for the dataset to be able to compose subsequent *AccessData* requests.

- **FITS**. FITS is the standard data format for returning image datasets from a service. The serialization used depends upon the complexity of the image dataset, and in particular on whether it is sparse. A "simple image" instance (metadata plus a single Data element) can be serialized as a simple FITS image. Sparse images are serialized in multi-extension FITS format. Details on the FITS serializations are given in the ImageDM specification.

Additional image data formats may optionally be supported by a SIA service instance, but are not currently defined by SIAP or the ImageDM. Generic formats include the Hierarchical Data Format (**HDF**), a generic data container capable of storing image data [ref], and **JPEG2000**, a multi-resolution (wavelet encoded), dynamically streamable, multi-dimensional image-encoding [ref]. Optional support for custom framework-specific data formats such as **CASA image tables** [ref] and **Starlink NDF** [ref], is also possible to optimize the data flow from SIA service instances to a particular data analysis system. Although not a data format to be returned from a SIA service, Image instances or their underlying components can also be mapped onto tables in a database management system (**RDBMS**).

2 Interface Summary

The basic form of a SIA service is defined by the DAL interface standard (DALI). A SIA service defines the following *resources* [DALI-2]:

Resource	***Type***	***Description***
/sync	sync	Used to execute synchronous requests.
/async	async	Used to execute asynchronous requests.
/availability	sync	VOSI resource that reports whether the service is available and able to respond to requests.
/capabilities	sync	VOSI resource that returns an XML document describing the capabilities of the service.

In addition the following *service requests* (service operations) are defined:

Request	***Sync***	***Async***	***Description***
QueryData	MAN	OPT	Query the service to discover available image datasets matching the given query constraints. The response is a table listing all available datasets. Full metadata is returned describing each image dataset. The described datasets may be either static ***archival*** datasets (pre-existing datasets in the remote archive), or ***virtual data*** datasets more closely matching the client request, to be computed and returned upon request by the client. An **access reference URL** is returned for each dataset that may be used to directly

			download the dataset.
AccessData	OPT	OPT	Directly access a single image dataset, returning the requested view of the data (usually a small subset, e.g., a filtered product, cutout or projection). AccessData by its nature normally computes and returns virtual data.
StageData	n/a	OPT	Request asynchronous computation and staging of data products (often virtual data) referenced in a prior QueryData operation.

Mandatory service elements are indicated as MAN and optional elements as OPT. Service requests are invoked by submitting the request to either the /sync or /async resource. A complete HTTP request to access the service is formed by appending the resource path and request name (if appropriate) to the *baseurl* for the service. For example, suppose the baseurl of a service is "http://example.com/base"; the corresponding synchronous queryData request might be as follows (this is all case-insensitive except possibly for specific parameter values):

http://example.com/base/sync?REQUEST=querydata&POS=12.1,0.0&SIZE=0.2

The simplest possible SIA service would implement only a synchronous *queryData* request, returning a description of static archival image datasets matching the query parameters, with an access reference URL provided for each dataset to retrieve the dataset. This may be sufficient for a uniform data collection of moderate sized images. Support for virtual data generation is desirable to provide more efficient and scalable data access for distributed analysis, and is usually required to access very large data products that are impractical to download.

2.1 Synchronous Requests

Synchronous requests must be submitted to the /sync service resource. Synchronous requests execute immediately, returning the response specified by the request directly to the client. For example, a synchronous *queryData* request will return a VOTable wherein each row of the table describes an available (archival or virtual) Image dataset matching the query constraints, while *accessData* will (usually) directly return an Image dataset as the request response.

2.2 Asynchronous Requests

Asynchronous requests must be submitted via an HTTP POST to the /async resource for the service. Asynchronous requests initiate a batch job to execute the request, returning an immediate response referencing a job description for the submitted job. The job description URL may be used to monitor and control subsequent job execution. Depending upon the capabilities of the specific service, multiple jobs may be submitted to execute concurrently. Further information on how to submit and manage asynchronous jobs is given in the DAL interface standard [DALI-2.1], following the Universal Worker Service design pattern [UWS].

2.3 Errors and Exceptions

SIA protocol errors (e.g., an invalid request) must return a VOTable error document [DALI-4] with QUERY_STATUS set to ERROR. Successful requests that return a VOTable response must set QUERY_STATUS to OK. If overflow occurs (for example in a discovery query where many candidate datasets are found), QUERY_STATUS is set to OVERFLOW; a valid VOTable response is otherwise returned containing the maximum number of records [DALI-4.4.1]. Service level errors (e.g. resource not found) or other errors at the HTTP protocol level must respond with the appropriate HTTP error code.

3 Service Requests

3.1 QueryData

The purpose of the *queryData* operation is to find images (physical or virtual) that satisfy the specified query constraints. The response is a VOTable document wherein each row of the table describes one candidate image dataset. Referenced datasets may be directly downloaded via the given *access reference* URL, or may be passed on to the *stageData* operation for computation as an asynchronous job.

3.1.1 Query Mode and Virtual Data

The *queryData* request is used to find image datasets matching the client-supplied query constraints. This sounds simple enough, but the semantics of queryData become subtler when used to query for virtual data products. When querying for ***virtual data***, the data products described in the query response will be virtual (non-physical or unrealized) data products that will be generated only if requested or accessed by the client. A typical example of an image virtual data product is a cutout or reprojection of a region of a larger image, although other types of dynamic image generation such as generation of an output image in a client-specified data format are also possible.

The queryData request provides an ***automated virtual data generation*** capability, relying upon each data service to determine the best match of the locally available data to the client request. This is necessary to allow the same request to be broadcast to multiple services, is necessary for scalability to enable global data discovery, and allows data discovery and optimization of virtual data products to be performed in a single operation (the *accessData* request may be used instead if more explicit, precision client-directed access to a single dataset is desired).

The ***query mode*** may optionally be specified in a queryData request to tell the service whether or not virtual data generation is desired, and is so, what level of manipulation of the data samples is permitted.

Query Mode	***Description***
archival	Discovery is limited to entire archival image datasets.

cutout	Allow the service to compute virtual data images that filter or exclude data to most closely match the multi-dimensional coverage specified by the query parameters. No resampling is permitted; the original pixels or voxels are returned without change. The coverage of the computed data product may only approximate the query region, but the amount of data to be retrieved may be greatly reduced.
match	The service is allowed full flexibility to compute a virtual data image most closely matching the query parameters, including producing a cutout, or resampling, reprojecting, or otherwise manipulating the original data so as to produce the best possible match to what was requested.

Specification of an explicit query mode is optional, and by default the service should compute (a description of) the best match to the client request possible, i.e., the query mode defaults to “match” or “cutout” for services that have this capability, otherwise “archival”. If an explicit query mode is specified it is a request to the service to “discover” the requested type of data product if possible for the particular data collection or dataset being queried.

If the query mode is “archival” the service should describe all archival (whole) image datasets matching the query. If the query mode is “cutout” the service should describe all virtual image datasets that will be filtered or trimmed along each image axis to fit the ROI without otherwise affecting the data samples. If the service cannot generate cutouts it is not an error, but the service should not “discover” any image cutouts (hence a query for image cutouts posed to a service that can only discover archival images will succeed but will return zero results). The “match” query mode is like “cutout”, but the service is requested to compute the best match to the query possible for the service, including reprojecting or otherwise resampling the data as necessary.

The client may request ***multiple query modes*** within a single query. The query mode “archival” will only find archival image datasets. The mode “cutout” will only find non-resampled image cutouts. The mode “match” will perform a best match extraction, falling back to cutout generation if that the best match the service is able to generate. The modes “archival,cutout”, or “archival,match” would find both archival image datasets as well as extracted virtual data subimages at the level indicated. If both “cutout” and “match” are specified in the same query, this is equivalent to a single “match” query. The *CreationType* attribute in the query response indicates how a particular image dataset is derived from the original data.

How certain ***query parameters*** are used in a query may depend upon the query mode, as specified in the description of each query parameter. In particular the meaning of the POS and SIZE parameters, specifying the spatial ***region of interest*** (ROI), depends upon the query mode. In a simple discovery query, searching for archival image datasets, POS and SIZE function much like a cone search, finding images that

approximately overlap the specified search region. If virtual data generation is enabled, POS and SIZE instead specify the footprint of the ***ideal image***, and the service will compute the virtual data product most closely matching the specified query parameters; if full resampling is possible the computed image may exactly match what was requested. The detailed semantics of parameters that have query mode dependence are specified in the descriptions of the individual parameters.

All query constraint parameters pertaining to the multi-dimensional coverage of a dataset can play a role in generation of virtual data products. In particular, the POS, SIZE, BAND, TIME, and POL parameters may be used to filter or constrain the data along each of the physical measurement axes. If resampling is enabled then the output image may further be reprojected to align with the specified ROI. The optional REGION parameter provides an alternative to POS, SIZE, BAND, TIME, and POL to specify a more complex multi-dimensional region to be used for data discovery or virtual data (e.g., cutout) specification.

3.1.2 Query Constraints

A data discovery query uses parameters as query constraints to find only data of interest to the client. Other parameters affect the operation of the service but not the selection of data.

Mandatory constraint parameters are parameters that a service is required to implement and use to constrain the query, if present in the query submitted to the service (these parameters are mandatory for the service to support, but the client is not required to use them in a query). Examples of mandatory query constraint parameters are POS, SIZE, BAND, TIME, POL, and FORMAT. Services are required to support these minimum query constraints in order to allow selection of datasets based upon any combination of constraints specifying the physical coverage or format of the dataset.

Additional ***optional*** constraint parameters may be used to further refine the query. If the service does not support an optional constraint parameter defined by the protocol it must permit the parameter to be present in the query without error, even if the parameter is not actually used as a query constraint by the service. If a constraint parameter is not included in the query or is not supported by the service, or cannot be applied to the data due to insufficient dataset metadata (note this is different than the case of theory data described below), then the parameter constraint is not applied. This allows a query to succeed even if it includes parameter constraints that the service instance does not support, so that the same query can be submitted to multiple service instances when performing global data discovery. Since discovery queries can be imprecise it is up to the client to analyze the query response to further refine the query.

If a service supports a parameter but the value given cannot be parsed or is otherwise illegal (as opposed to merely not matching any data) then an ***error response*** should be returned to the client. If a service does not support a parameter it is not required to parse the parameter value and report errors, i.e., it may ignore the unsupported parameter.

Specific constraints may or may not have meaningful values for a given data collection. For example, for ***theory data***, anything having to do with time or position on the sky may be undefined. For solar or planetary data, time is defined but the spatial position on the celestial sphere may be undefined or not meaningful. In such a case, where a specific value is specified for an attribute which is undefined or has no meaning for a given data collection, the service should respond by finding no matching data (for example a query based on POS, if broadcast to a broad range of services, would probably not find any matching data if posed against a service providing access to theory data). For data collections where all physical measurement parameters are meaningful, for example spectra of galactic or extragalactic astronomical targets, all parameters should be supported and used to constrain the query, even if only imprecise values of the related attributes are known for a given dataset.

3.1.2.1 Mandatory Query Parameters

The following parameters must be implemented by a compliant service:

Parameter	*Typical value*	*Physical unit*	*Datatype*
POS	`52,-27.8`	degrees; defaults to ICRS	string
SIZE	`0.05`	degrees	double
BAND	`2.7E-7/0.13`	meters	string
TIME	`1998-05-21/1999`	ISO 8601 UTC	string
POL	`any`	-	string
FORMAT	`fits`	-	string

All services must support queries containing at least these six parameters, representing coverage in the fundamental physical measurement axes, and the output data format or formats desired by the client. Although services must support these parameters, this does not necessarily mean that the quantity referred to is meaningful for the class of data being queried. While a compliant service must implement these parameters and use them where possible to constrain queries, a valid client query can be composed from any combination of parameters, and may include or omit any given parameter. If a parameter is not specified, it is not used to constrain the query. For example if POS is not specified, data from any spatial region, or data for which POS is undefined, will satisfy the query and other parameters must be used to constrain the query.

3.1.2.2 POS

POS specifies the central position of the region of interest (ROI). The coordinate values are specified in list format (comma separated) with no embedded white space [SSA-8.7.2].

Example: `POS=52,-27.8`

POS defaults to right ascension and declination in decimal degrees in the ICRS coordinate system. The coordinate system may optionally be indicated to specify a coordinate system other than ICRS. When appropriate for the data being accessed the service must support the ICRS and GALACTIC coordinates systems (services for theory data and solar and planetary data for example would be an exception). The coordinate

system is specified as a list format modifier, with the acceptable values as defined in [XX].

Example: `POS=52,-27.8;GALACTIC`

POS may be used to reference spatial regions of any dimension. Spatial coordinates requiring more than two values (e.g., a simulation or reconstruction of a 3-D region of space) are possible merely by having more than two comma-delimited values before the qualifier.

Whether or not a service supports specific coordinate systems for POS is a service-defined optional capability. It is an error if a coordinate reference frame is specified which the service does not support.

[*A future version of SIAV2 may further generalize POS to support upload of multiple positions via the UPLOAD mechanism in DALI, to provide a multi-position search capability.]*

3.1.2.3 SIZE

SIZE specifies the dimensions (e.g., width and height) of the rectangular region of interest, defining either the ***ideal image*** footprint (for services operating in *cutout* or *match* mode where images are dynamically generated to match the given ROI) or the ***search region*** (for services operating in *archival* image search mode, searching for existing images which overlap the given ROI). A single ROI specified with POS and SIZE may be used to search for data regardless of the query mode.

The SIZE values define the full angular extent of the ROI in decimal degrees measured along a path parallel to the axes defined by the coordinate frame given (or implied) in the POS parameter and centered on the position give in the POS parameter. Since SIZE is specified as an angular extent the region size is constant regardless of position on the sky.

Example: `SIZE=0.05,0.03`

If only a single value is given it applies to both the width and height of the search region, otherwise the two values may be specified separately (values of SIZE with more than two values are also possible, as for POS).

In ***whole image*** discovery queries (archival query mode) a match occurs when a candidate image overlaps the specified ROI. The *n* SIZE values (*n*=2 for the celestial coordinate frames) define an *n*-dimensional modified cone search wherein the angular distance between the center of the ROI and the approximate *edge*[1] of the target image, measured along the coordinate axis, is tested separately for each axis of the spatial coordinate frame. The point of intersection with the edge of the target image is along a line from POS to the center of the target image, but may be approximated.

A special case is when SIZE=0 or is unspecified. For a service operating in archival image mode this tests whether the given position is in the image. For a service

[1] The edge test applies in the default case of INTERSECT="overlap". See the INTERSECT parameter (3.1.2.12) for additional options.

operating in cutout or match mode this will cause a small default-sized image cutout to be computed centered on the given position. The default image cutout size is service-defined and may be a value considered appropriate for the service, for the given image or data collection being accessed, or for the object (if any) at the given position.

The optional parameters REGION and INTERSECT (discussed in section 3.1.2.9) provide additional control of the search or extraction region.

3.1.2.4 BAND

The spectral bandpass is specified as an *ordered range-list* (SSA-8.7.2) either numerically as a wavelength value or range, or textually as a spectral bandpass identifier, e.g., a filter or instrumental bandpass name. If a single numerical value is specified it matches any dataset for which the spectral coverage includes the specified value. If a two-valued range is given, a dataset matches if any portion of it overlaps the given spectral region. The range list may contain multiple elements in which case a candidate dataset matches if it matches any element of the range list.

For a numerical bandpass the units are wavelength in vacuum in meters in the rest frame of the source.

If a bandpass is specified as a string identifier it is assumed to be a bandpass identifier such as a standard VO bandpass name (“radio”, “optical”, “x-ray”, etc.) as specified in the VO resource metadata [23] for *Coverage.Spectral*. Instrumental bandpass names may also be used but are not well defined; allowable values may be collection-specific and can often be determined by an initial query to the data service. A list of bandpass identifiers is permitted, but numerical and string bandpass values may not be mixed within the same range-list.

[*Filtering the spectral axis can require an arbitrarily long list of spectral ranges, used to either include or exclude spectral regions*. *A future version of SIAV2 may further generalize BAND to support upload of lengthy range-lists via the UPLOAD mechanism in DALI.]*

3.1.2.5 TIME

The temporal coverage (epoch of observation) is specified as an *ordered range-list* (SSA-8.7.2) of time values or ranges. Time values and ranges are specified in ISO8601 format [DALI-3.1.2]. If the time system used is not specified UTC is assumed. The value specified may be a single value or an open or closed range. If a single value is specified it matches any dataset for which the time coverage includes the specified value. If a two-valued range is given, a dataset matches if any portion of it overlaps the given temporal region.

[*Filtering the time axis can require an arbitrarily long list of time ranges, used to either include or exclude time regions*. *A future version of SIAV2 may further generalize TIME to support upload of lengthy range-lists via the UPLOAD mechanism in DALI.]*

3.1.2.6 POL

Specifies whether or not data is desired which measures polarization, and if so the type of polarization desired, specified as a single value or as a *list* [SSA-8.7.2]. The default is to ignore polarization during data discovery. Possible values include the following:

POL Value	***Description***
any	Find only datasets that measure some form of polarization.
none	Find only datasets that do not measure polarization.
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

The values “any” and “none” may be used to discover datasets that explicitly do or do not measure polarization of any form (a dataset that only provides Stokes I is not considered to include a polarization measure). The Stokes parameters I, Q, U, V may be used to filter datasets that represent polarization using the Stokes parameters. Additional forms of polarization (RR, LL, RL, XY, etc., or other custom measures) may be selected if present in the data to be queried. For computation of derived values such as the polarization intensity and angle see *accessData* (3.2).

[*Another possibility would be to discover data based upon the percent polarization, although this is not well-defined, e.g., linear or circular polarization have different measures, and it is more a property of a specific target than the overall image*.]

3.1.2.7 FORMAT

The FORMAT parameter specifies the allowable data formats for a discovered image. The value is a comma-delimited *list* [SSA-8.7.2], wherein each element may be any recognized MIME-type or a special reserved value, for example:

```
fits, image/jpeg, text/html
```

In addition to the open set of standard MIME-type format specifications, the following special shorthand values are defined:

FORMAT	***Meaning***
`all`	All formats supported by the service (default).
`fits`	Shorthand for `image/fits` or `application/fits`
`graphic`	Any of the graphics formats: JPEG, PNG, GIF.
`html`	The image is rendered as a HTML page with annotations.
`metadata`	Return only query response metadata (3.1.2.8)

If FORMAT is omitted, “all” should be assumed, and the service should describe all available formats. FORMAT values are case insensitive.

The FORMAT parameter describes the desired format of returned image data. If no data is available for any of the requested formats, a null query response should be

returned indicating that no data satisfying the query is available (this is equivalent to FORMAT = "metadata"). If the query response describes virtual data, the service may generate data in the format requested by the client, on the fly when the data is subsequently accessed. Note FORMAT applies only to the data; the query response itself is always returned as a VOTable.

[*Additional format shortnames for image formats such as "html5", "jpeg2000", etc. are possible, but are probably best left to MIME types at present. A possible enhancement would be to add support for RESPONSE_FORMAT, to allow the query response to be returned in formats other than VOTable, e.g., csv, tsv, json, and so forth, as are useful for some applications. A complication is that this might need to be combined with a SELECT-like capability to limit the amount of metadata* returned.]

3.1.2.8 Query Response Metadata

Since much of the ImageDM metadata is optional, different service instances may return different subsets of metadata in a queryData response, possibly augmented with some custom metadata attributes. A query to determine only the *metadata* to be returned in a discovery query may be invoked via a ***null query***, that is, a valid query that finds no data but returns a valid query response VOTable. This may be achieved by issuing a queryData request in either of two ways, either with FORMAT set to "metadata" (an explicit queryData metadata query) or with MAXREC set to zero (an explicit null query).

3.1.2.9 Recommended and Optional Query Parameters

The following additional parameters may be implemented by a service; at least the "recommended" parameters should be implemented by more advanced services that provide more than the most basic capabilities. In the table below, recommended parameters by REC, and optional parameters by OPT.

Parameter	*Sample value*	*Unit*	*Req*	*Datatype*
MODE	cutout		OPT	string
REGION	Circle ICRS 148.9 69.1 2.0		OPT	string
INTERSECT	center		REC	string
SPECRES/RP	2000	λ, λ/δλ	OPT	double
SPATRES	0.05	degrees	REC	double
TIMERES	31536000 (=1yr)	seconds	OPT	double
FLUXLIMIT	?	?	OPT	double
TARGETNAME	mars		REC	string
TARGETCLASS	agn		OPT	string
ASTCALIB	absolute		OPT	string
FLUXCALIB	relative		OPT	string
TYPE	cube		REC	string
SUBTYPE	nrao.vla.cube		OPT	string
PUBDID	ADS/col#R5983		REC	string

CREATORDID	ivo://auth/col#R1234		OPT	string
COLLECTION	SDSS-DR7		OPT	string
TOP	20	none	OPT	int
MAXREC	5000		REC	string
MTIME	2005-01-01/2006-01-01	ISO 8601	OPT	string
COMPRESS	hcompress		OPT	string
RUNID			REC	string

The spatial, spectral, and time resolution of the data may all be used as query constraints to find data of interest, or to aid in the generation of virtual data products. The flux limit specifies the minimum sensitivity of data required for analysis. The target name and class may be used to search for data for a specific target, or for a specific type of astronomical object. The creator and publisher dataset identifiers and data collection name may be used to specify the individual dataset or data collection to be accessed. TOP and MAXREC are used to manage the queryData response returned to the client. All parameters are explained in more detail below.

3.1.2.10 MODE

The query mode as specified in section 3.1.1: one of “archival”, “cutout”, or “match”. The default if not specified is for the service to specify the data product most closely matching what the client requested, that the service is able to provide. The value is a single value or a list of type string (SSA-8.7.2) specifying the levels of data products to be discovered and specified in the query response.

3.1.2.11 REGION

The REGION parameter provides an alternative to POS, SIZE, BAND, and TIME to specify a region in multi-dimensional space as an STC-S formatted string [XX, 22]. In *archival* query mode candidate images match which overlap the specified region. In *cutout* or *match* mode images will be computed that provide a best match to the specified region. REGION may be combined with the resolution parameters to specify the resolution, along any image axis, of discovered or generated images.

REGION may specify any combination of spatial, spectral, and time coverage (polarization is omitted as this is not currently supported by STC-S). If any of POS, SIZE, BAND, TIME, and POL are specified along with REGION then all such constraints must be satisfied. For example, REGION might specify some large region of multi-dimensional space, and any combination of the other parameters may be used to more finely filter the data within the specified region.

When REGION is used the search region is no longer limited to non-rotated rectangles, e.g., circles, ellipses, rotated regions, and arbitrary polygons may be specified as well.

Example: `Circle ICRS 148.9 69.1 2.0`

A service which implements REGION must support at least the ICRS and GALACTIC spatial coordinate frames, if appropriate for the data being accessed. Unlike most

optional query constraints it is an error if the REGION parameter is specified but is not supported by the service, because it can substitute for the mandatory parameters.

3.1.2.12 INTERSECT

A parameter that indicates how matched images should intersect the region of interest. The allowed values are:

* CONTAINS – The ROI fully covers the candidate image.
* COVERS – The candidate image fully covers the ROI.
* CENTER -- The candidate image overlaps the center of the ROI.
* OVERLAPS -- The candidate image overlaps some part of the ROI.

INTERSECT applies to regions specified by POS, SIZE and/or REGION in all query modes. If this parameter is not specified, INTERSECT="overlaps" is assumed.

The default value "overlaps" requires that target images overlap the ROI hence have some coverage within the ROI. The value "center" requires additional overlap, with coverage including the center of the ROI. The value "covers" requires that the target image fully cover the ROI. The value "contains" requires that the candidate image be fully contained within the ROI.

The INTERSECT constraint may be computed as an n-D modified cone search. For each of the *n* axes the angular distance between the center of the ROI and the center of the target image, projected to the measurement axis, is computed. The four intersect conditions then reduce to different combinations of the half-width (along the measurement axis) of the ROI and target image rectangles (in particular, "contains" and "covers" are the inverse of each other, swapping the ROI and target image). Approximations may be used to avoid computing exact degrees of overlap.

3.1.2.13 SPECRES, SPECRP

The minimum spectral resolution, specified as either the minimum spectral resolution specified as a wavelength in meters (SPECRES), or as the spectral resolving power $\lambda/d\lambda$ in dimensionless units (SPECRP). For a spectral data cube SPECRES and SPECRP refer to the spectral axis of the cube. For a 2-D image the spectral resolution corresponds to the bounds of the bandpass of the image, and the spectral resolving power is unity.

3.1.2.14 SPATRES

The minimum spatial resolution specified in decimal degrees. Spatial resolution refers to the PSF of the observed signal and is independent of the pixel size of the image so long as the image is adequately sampled.

3.1.2.15 TIMERES

The minimum time resolution specified in seconds. For a time cube the time resolution refers to the time axis of the image. For a 2-D image the time resolution corresponds to the bounds of the time coverage of the exposure.

3.1.2.16 FLUXLIMIT

The maximum allowable RMS noise level specified in microJy. Smaller values correspond to more sensitive datasets with a fainter detection limit.

3.1.2.17 TARGETNAME

The name of the target (astronomical object) observed by a dataset. Data discovery is normally performed by using an external name resolver (e.g., NED, SIMBAD) to get the coordinates of an astronomical object, which are then input to queryData using POS. However when searching for data for which position is meaningless or not well defined, e.g., observations of solar system objects, the target name must be used for discovery instead of position. Services that provide such data should implement TARGETNAME and allow it to be used for discovery.

3.1.2.18 TARGETCLASS

A comma delimited list of strings denoting the classes of astronomical objects to be searched for.

Examples: `star, galaxy, pulsar, PN, AGN, QSO, GRB`

This corresponds to *Target.Name* in the dataset metadata for a candidate dataset.

3.1.2.19 ASTCALIB

Specifies the minimum level of astrometric (spatial) calibration required. Possible values are “`absolute`”, "`relative`", and "`any`" (the default). Datasets with an accurate WCS have an “absolute” astrometric calibration. A discovery search specifying ASTCALIB as relative or absolute will exclude image datasets that do not include a WCS.

3.1.2.20 FLUXCALIB

Specifies the minimum level of flux calibration for acceptable data. Possible values are “`absolute`”, "`relative`", and “`any`” (the default). If "relative" is specified, datasets that have a higher level absolute flux calibration will be found as well. A SED builder application for example, would want to search for image datasets with an absolute flux calibration.

3.1.2.21 TYPE

TYPE specifies the data product (dataset) type as defined in ObsTAP [12]. For SIA the possible dataset types as specified in ObsTAP are always “image” or “cube” (the latter for image datasets of dimension greater than 2). Hence, TYPE may be used to find only image data of dimension 2 or less, or only image cube data with 3 or more physical (as opposed to WCS) dimensions.

3.1.2.22 SUBTYPE

SUBTYPE specifies the dataset subtype as defined in ObsTAP [12]. The subtype is the specific data product type as defined in terms of an individual archive or data collection.

For example, if a number of data products are available for an instrumental data collection, each should be assigned a unique subtype identifier to make it trivial to query for a specific type of data product.

3.1.2.23 PUBDID

The IVOA publisher's dataset identifier, assigned by the publisher of a dataset. Setting PUBDID in a queryData request limits the operation to a single dataset and may be used to retrieve metadata for that specific dataset.

PUBDID will uniquely identify a dataset within the collection managed by the publisher, however the same dataset published in different places may have a different PUBDID assigned by each publisher. This differs from CREATORDID that is globally unique for a given dataset; however it is common for dataset creators to fail to assign unique IDs to newly created data. A data publisher can always assign a unique PUBDID when a dataset is published to the VO. ADS dataset identifiers are an example of a PUBDID, but in general any publisher may assign their own unique publisher dataset identifier. Publisher dataset identifiers may be determined by a prior query or some external means, such as another form of archive query.

Note:
A special case of a publisher's dataset identifier is the **ADS dataset identifier,** used to reference published IVOA datasets in journal articles.

3.1.2.24 CREATORDID

An IVOA dataset identifier, assigned at creation time by the creator of the parent data collection (survey project, observatory, etc.). Datasets may have a globally unique CreatorDID assigned prior to publication of the data to the VO, for example when the data is generated by a processing pipeline, or ingested into the master archive for the data collection. CreatorDIDs are globally unique since the Creator entity for a data collection (e.g., an observatory or survey project) controls its own namespace, which can be registered with the IVOA as a globally unique Authority identifier. When a CreatorDID has been assigned this is the most universal way to refer to a dataset, as all replicated versions will share the same CreatorDID regardless of where they are published. Creator dataset identifiers may be determined by a prior query or by some other means, such as another form of archive query.

Example: `ivo://nrao.edu/vla#1998s2/4992a`

3.1.2.25 COLLECTION

COLLECTION specifies either the IVOA resource identifier or the "*shortname*" of a data collection as defined by the service, for example `ivo://nrao.edu/vla`, or `SDSS-DR7`. By data collection we refer to an organized, uniform collection of datasets from a single source, for example a single data release from a survey, or an instrumental data collection from an observatory. Unless an IVOA identifier is input, the service should treat the search term as a case insensitive, minimum match string. For instance, "dss"

would match either `dss1` or `ESO-DSS2`. The defined data collection references are specified in the service capabilities for a particular data collection.

3.1.2.26 TOP

TOP limits the number of returned records in the query response table to the specified number of "top ranked" ones. Records are ranked according to a "score" heuristic. The details of the actual heuristic used are up to the service, but the general idea is that the better a candidate dataset matches the query, the higher the score it receives. Metrics such as distance from the specified position, or the degree of overlap with a specified spectral bandpass or time interval, determine the score. If two datasets would otherwise have the same score, the service may use other unspecified dataset characteristics, such as some intrinsic data quality metric, to further rank candidate datasets. If the service implements a ranking heuristic the query response table should normally be returned sorted in order of decreasing score.

TOP can also be used by the client to limit the size of the query response table in cases where the query might find a very large number of candidate objects. This is similar to MAXREC, but TOP forces ordering of the query response, and does not indicate overflow.

3.1.2.27 MAXREC

The maximum number of records to be returned in the query response. MAXREC [DALI-3.2.4] may be used by the client to increase or decrease the built-in default query size limit defined by the service, up to some maximum service-specified default. A service should typically have a modest default MAXREC, providing a reasonable query response time, and a large upper limit on MAXREC, provided to enable large queries. Very large values of MAXREC may allow streaming an arbitrary amount of data back to the client.

If a query response exceeds the value of MAXREC currently in effect, MAXREC rows of data should be returned to the client, setting the query status value to OVERFLOW to indicate that overflow occurred. It is not an error if query overflow occurs.

3.1.2.28 MTIME

Find only datasets modified, created, or deleted within the given range of dates, specified as a single element in range-list format [SSA-8.7.2], as an open or closed range, with the dates specified in ISO 8601 format [DALI-3.1.2]. Note this is not the same thing as TIME, which refers to time of observation. MTIME may be used to periodically query services for new or updated data. Deleted datasets are indicated by a non-null deletion date in the *Dataset.Deleted* field of the query response. Services that support MTIME should also support *Dataset.Deleted*.

3.1.2.29 COMPRESS

If this flag is present, datasets retrieved via the access reference URL may optionally be returned to the client in a standard compressed format determined by the service.

Clients must be capable of handling a range of standard astronomical image compression schemes. By default compressed data is not permitted.

Dataset-level compression is distinguished from protocol-level compression, which is performed at the level of the HTTP protocol, on the entire data stream, and is transparent to the client.

[*We may need additional control to specify the level of compression. Lossy compression could also be supported*.]

3.1.2.30 RUNID

The RUNID [DALI-3.2.6] is an opaque string used to associate multiple service invocations in service logs, e.g., to identify them as all belonging to the same job or application. RUNID is not used by SIA itself, except in cases where SIA may call another VO service, in which case the RUNID parameter should be passed on to the called service. The purpose of RUNID is to allow the job run ID to be logged, and in particular, if a job involves multiple requests to multiple services, to allow all just requests to be associated by having a common RUNID. This applies to all service requests regardless of whether they execute synchronously or asynchronously.

3.1.3 Service-Defined Parameters

The service may support additional service-defined parameters. Parameter names must not match any of the reserved parameter names defined herein, independent of case.

Any service-defined parameters should be defined in the capability metadata for the service [DALI-2.3.2]. Clients may examine this metadata to retrieve the definitions of the custom service-defined parameters, e.g., to auto-generate a query form for the service. Typical examples of the use of custom query parameters are extensions by a service to customize the query for a particular data collection, or services used to query theoretical models, where the model parameters are input with custom parameters.

3.1.4 Successful QueryData Response

The output returned by a query is an XML document compliant with VOTable V1.2 or greater (VOTable 2009) and must be returned with a base MIME-type of `text/xml`. It is recommended that the reported MIME-type be further refined "`text/xml;content=x-votable`" to indicate the response is a VOTable.

Note:
The `FORMAT` parameter has no influence on the query response. `FORMAT` applies only to the returned datasets, not to the query response. The

query response is always returned as a VOTable.

The VOTable must contain a RESOURCE element, identified with the attribute type = "results", containing a single TABLE element with the results of the query. Additional RESOURCE elements may be present, and the usage of any such elements is defined below (extensions).

The RESOURCE element must contain at least two INFO elements (as a direct childen). One must have the attribute name="QUERY_STATUS"; its value attribute should be set to "OK" if the query executed successfully, regardless of whether any matching data were found. All other possible values for the value attribute are described below (section 3.1.5). The element may contain as its content a message appropriate for display indicating a successful result.

Examples:

```
<INFO name="QUERY_STATUS" value="OK"/>
<INFO name="QUERY_STATUS" value="OK">Successful Search</INFO>
```

The other required INFO element must have the attributes name="SERVICE_PROTOCOL" and value="2.0". Its content should contain for display purposes the string, "SIA2".

Example:

```
<INFO name="SERVICE_PROTOCOL" value="2.0">SIAV2</INFO>
```

Additional INFOs may be provided. The recommend way to include the input parameters that were used to generate the results is through additional INFO elements: with each such INFO element, the name attribute should be the name of the input parameter prepended with the string, "INPUT:" The value attribute should contain the value of the parameter provided.

Example:

```
<INFO name="INPUT:POS" value="183.25667,-13.4456"/>
```

In the query response table each row represents a different physical or virtual dataset which is potentially available to the client. Table FIELD elements enumerate the image metadata encoded in the columns of the table and which describe each ; if all datasets share the same value for a metadatum, it can be represented as a PARAM. The VOTable GROUP construct is used to associate related groups of fields. The next section describes the details for describing the image metadata.

3.1.5 Query Response Metadata

The FIELD and PARAM elements identify and describe the metadata returned which describe the images matching the input query. Apart from metadata that are specified as required or recommended by this document, the metadata that are returned is the choice of the data provider; generally, this should be metadata sufficient to enable the user to understand what each image represents and decide which ones (or portions thereof) should be downloaded. The Image Data Model (ref) reflects metadata considered important for sufficiently describing an image. The names of the FIELD and PARAM elements—i.e. the value provided via their name attributes—is the choice of the provider; typically, these are the names provided to users through the repository's native interfaces. Similarly, this document places no restrictions on the values of the id attribute apart from what is required by the VOTable specification. This document places no restriction on the order of the columns (as indicated by the order of the FIELD elements) apart from the rules specified by the VOTable specification in the use of the GROUP elements (i.e. that fields that are part of a common group must appear consecutively).

The utype attributes of the FIELD and PARAM elements (and, where appropriate, of any GROUP elements) are set to indicate what the metadata semantically represent. Except where specified otherwise by this document, the values of the utype attributes should be chosen from the UType identifiers defined in the Image Data Model (ref). These identifiers must be prepended with namespace string "im" to indicate that they are UTypes of the Image Data Model.

Example:

```
<FIELD name="imname" utype="im:DatasetTitle" type="char"
       arraysize="*" />
```

Providers are encouraged to provide a utype attribute be provided to every FIELD, PARAM, and GROUP elements unless there is no UType whose meaning matches that of the metadatum.

Providers may set the values of the FIELD and PARAM elements' ucd attributes. This document recommends particular UCD values for fields labeled with certain Image Data Mode UTypes.

3.1.5.1 Required Columns

The query response table must include exactly one FIELD or PARAM element with a utype attribute assigned with each of the following UType identifiers which are defined in the Image Data Model [ref].

UType Identifier	datatype	arraysize	unit
DataID.Title	char	*	
Dataset.Image.Nsubarrays	int	1	

Dataset.Image.Naxes	int	1
Dataset.Image.Naxis	int	*
Dataset.Image.WCSAxes	Char	*
Access.Format	char	*

The datatype, arraysize and unit columns above indicate the values of the `datatype`, `arraysize`, and `unit` attributes which must be set on the `FIELD` or `PARAM` element tagged with the corresponding UType. The values for these metadata must be given in the units specified in the units column. An unspecified unit (as in for all of the metadata in this particular set) indicates that a unit is not applicable; consequently, the unit attribute should not be set. An arraysize value of 1 is the default value for the `arraysize` attribute; in this case, the `arraysize` attribute need not be set. Otherwise, the metadata value is an array of numbers of the corresponding shape according to the arraysize value rules defined in the VOTable specification [ref]. Columns with type="char" and arraysize="*" simply contain string values.

Values for the column having the UType, Dataset.Image.WCSAxes, are formatted as a space-delimited list of words in which each word is a label indicating the type of coverage provided by an axis of the image. The order of the labels must correspond to the order of the axes in the data array. There is no controlled list of labels specified by this document; however, the label should be indicative of the type to the human reader. In particular, it is recommended that, where applicable, labels are used that correspond to those commonly used as values for FITS keywords CTYPE* after dropping the suffix representing the mapping algorithm (e.g. projection).

Example:
Recommended: `<TD>RA DEC VELO STOKES</TD>`
Not Recommended: `<TD>RA--SIN DEC-SIN VELO-HEL STOKES</TD>`

3.1.5.2 Recommended Columns for Physical Coverage

In general, the response table should contain columns that describe the axes of the image. In the case of a repository that contains images with axes which provide physical spatial and spectral coverage (e.g. as with real observations of the sky), it is *strongly recommended* that the response table contain columns with the following UTypes:

UType Identifier	datatype	arraysize	unit
Char.SpatialAxis.Coverage.Location.coord	double or float	2	deg
Char.SpatialAxis.Coverage.Bounds.Limits.LoLimit2Vec	double or float	2	deg

Char.SpatialAxis.Coverage.Bounds.Limits.HiLimit2Vec	`double` or `float`	2	`deg`
Char.SpatialAxis.Resolution.RefVal	`double` or `float`	1	`deg`
Char.SpectralAxis.Coverage.Location.Coord	`double` or `float`	1	`m`
Char.SpectralAxis.Coverage.Bounds.Limits.LoLimit	`double` or `float`	1	`m`
Char.SpectralAxis.Coverage.Bounds.Limits.HiLimit	`double` or `float`	1	`m`
Char.SpectralAxis.Resolution.RefVal	`double` or `float`	1	`m`

The columns in the table above should be interpreted as described in section 3.1.5.1. For these metadata, the `datatype` attribute may be set to either `"double"` or `"float"`, depending on the format or precision of these values as stored natively by the repository. Values in the column tagged with the UType, Char.SpatialAxis.Coverage.Location.coord, must correspond to the ICRS position of the approximate center of the image, given as right ascension and declination in decimal degrees. Values in the column tagged with the UType, Char.SpectralAxis.Coverage.Location.coord, must correspond to the approximate center of the waveband covered by the image.

When there are images in the repository in which the sampling over time is well defined and known, the response table should provide time coverage metadata. In particular, if the service can apply the TIME query parameter constraint when selecting result records, it is *strongly recommended* that the results table should include columns with the following UTypes:

UType Identifier	`datatype`	`arraysize`	`unit`
Char.TimelAxis.Coverage.Location.coord	`double` or `float`	`1`	`d`
Char.TimelAxis.Coverage.Bounds.Limits.LoLimit	`double` or `float`	`1`	`d`
Char.TimelAxis.Coverage.Bounds.Limits.HiLimit	`double` or `float`	`1`	`d`
Char.TimeAxis.Resolution.RefVal	`double` or `float`	`1`	`s`

When the repository includes images that explicitly sample different polarizations, the response table should include columns that describe the polarization measurements available in an image dataset. In particular, if the service can apply the POL query parameter constraint when selecting result records, it is *strongly recommended* that the results table should include columns with the following UTypes:

UType Identifier	`datatype`	`arraysize`	`unit`
Char.PolAxis.Enumeration	`char`	`*`	

A value in the column tagged with the UType, Char.PolAxis.Enumeration, is a string space-delimited list of polarization names; allows names are as follows:

Pol. name	Definition
I	Stokes I
Q	Stokes Q

Pol. name	Definition
U	Stokes I
V	Stokes V

Pol. name	Definition
R	Right-handed circular
L	Left-handed circular

If a particular image does not have coverage over one of the above coverage domains or if the coverage is otherwise unknown, the corresponding metadata values should be specified as null according to the VOTable specification. (Note that if a service supports null values in the response table, it should use VOTable v1.3.)

Finally, to indicate what the values in the data array represent, it is recommended that query results include a column with the UTYPE, Char.FluxAxis.UCD:

UType Identifier	datatype	arraysize	unit
Char.FluxAxis.UCD	char	*	

The value for this column must comply with the UCD 1+ [ref] standard. When the dataset being described contains multiple data arrays (in which the quantity being measured may be different, as in magnitude versus error), the column value should be the UCD corresponding to the primary data array.

3.1.5.3 Recommended Columns for Data Access

If any of the images that can be described in rows of a QueryData response table is wholly available for download, then it is *strongly recommended* that the response table should include columns with the following UTypes:

UType Identifier	datatype	arraysize	unit
Access.Reference	char	*	
Access.Size	long	1	kbyte

If the service only provides access to portions of image datasets via the AccessData operation, then these columns need not be provided. In the case where these columns are provided in the response table but where particular images listed are not available for download, the values for these metadata should be set to null (according to the VOTable specification).

3.2 AccessData

3.2.1 AccessData Request

3.2.2 AccessData Response

3.3 StageData

3.3.1 StageData Request

3.3.2 StageData Response

Appendix A: VOTable Serialization

Insert appendix here

Appendix B: FITS Serializations

[*This should probably be moved to the ImageDM*.]

Appendix C: Change History

Changes since VAO Baseline, 20130812:

- S3.1.5: changed prefix for ImageDM UTypes to “im”
- S3.1.2.3: Clarified that Size represent the full height/width of the ROI
- S3.1.2.6: Added R,L as possible values
- S3.1.5.1: Added Dataset.Image.WCSAxes to list of required columns; specified format of values.
- S3.1.5.2: Coverage Bounds were added to the list of recommended columns describing coverage; this includes:
    - Char.SpatialAxis.Coverage.Bounds.Limits.LoLimit2Vec
    - Char.SpatialAxis.Coverage.Bounds.Limits.HiLimit2Vec
    - Char.SpectralAxis.Coverage.Bounds.Limits.LoLimit
    - Char.SpectralAxis.Coverage.Bounds.Limits.HiLimit
    - Char.TimeAxis.Coverage.Bounds.Limits.LoLimit
    - Char.TimeAxis.Coverage.Bounds.Limits.HiLimit
- S3.1.5.2: Added Char.FluxAxis.UCD as recommended with qualifications on the value being UCD 1+

References

[1] [**ImageDM**] D. Tody, F. Bonnarel, M. Cresitello-Dittmar, M. Louys, A. Rots, J. E. Ruiz, J. Salgado, *IVOA Image Data Model*, IVOA Working Draft, July 2013. http://wiki.ivoa.net/internal/IVOA/ImageDM/WD-ImageDM-20130505.pdf

[2] [**DALI**] P. Dowler, M. Demleitner, M. Taylor, D. Tody, DAL-WG, *IVOA Data Access Layer Interface,* IVOA Proposed Recommendation May 2013. http://www.ivoa.net/documents/DALI/

[3] [**SSA, PQL**] D. Tody, M. Dolensky, J. McDowell, F. Bonnarel, et. al., *Simple Spectral Access Protocol Version 1.1*, IVOA Recommendation, February 2012, http://www.ivoa.net/documents/SSA/

[4] [**VOSI**] M. Graham & G. Rixon (ed.), GWS-WG, *IVOA Support Interfaces Version 1.0*, IVOA Recommendation, 31 May 2011. http://www.ivoa.net/Documents/VOSI/

[5] [**UWS**] P. Harrison & G. Rixon, *Universal Worker Service Version 1.0*, IVOA Recommendation, 10 October 2010. http://www.ivoa.net/Documents/UWS/

[6] [**VOTable**] F. Ochsenbein (ed.), R. Williams, *VOTable Format Definition Version 1.2*, IVOA Recommendation 30 November 2009. http://www.ivoa.net/Documents/VOTable/1.2

[7] Tody, Plante, "*Simple Image Access Specification 1.0*", May 2009, http://www.ivoa.net/documents/latest/SIA.html

[8] Tody, Bonnarel, et. al, "*Simple Image Access Protocol Version 2.0*", IVOA Working Draft, Nov 2009 (predates the current SIAV2 draft), http://wiki.ivoa.net/internal/IVOA/SiaInterface/WD-SIAP-2.0-20091104.pdf

[9] D. Tody, A. Rots, B. Berriman, M. Cresitello-Dittmar, M. Graham, G. Greene, R. Hanisch, T. Jenness, J. Lazio, O. Laurino, R. Plante, *Access to Multidimensional (Cube) Data in the VO*, May 2013 http://wiki.ivoa.net/internal/IVOA/SiaInterface/CubeDataInVO.pdf

[10] D. Tody, F. Bonnarel, M. Dolensky, J. Salgado, DAL-WG, *IVOA Data Access Layer Service Architecture and Standard Profile,* IVOA Note 5 October 2008. http://www.ivoa.net/internal/IVOA/SiaInterface/DAL2_Architecture.pdf

[11] Michel, Bonnarel, Louys, "*IVOA Datalink Protocol*", http://www.ivoa.net/documents/Notes/DataLink/index.html

[12] Tody, Louys, Micol, Durand, et. al., "*Observation Data Model Core Components and its Implementation in the Table Access Protocol*", Oct 2011, http://www.ivoa.net/documents/ObsCore/20111028/index.html

[13] Bonnarel et. al., "*Data Model for Astronomical Dataset Characterisation*", March 2008, http://www.ivoa.net/documents/latest/CharacterisationDM.html

[14] Richards, "*Radio Interferometry Data in the VO*", Feb 2010, http://wiki.ivoa.net/internal/IVOA/SiaInterface/Anita-InterferometryVO.pdf

[15] Richards, Bonnarel, "*Note on the Description of Polarization Data*", May 2009, http://wiki.ivoa.net/internal/IVOA/SiaInterface/Polarization.pdf

[16] Greisen, Calabretta, "*Representations of World Coordinates in FITS*", 2002, http://arxiv.org/abs/astro-ph/0207407

[17] Calabretta, Greisen, "*Representations of Celestial Coordinates in FITS*", 2002, http://arxiv.org/abs/astro-ph/0207413

[18] Greisen, Calabretta, Valdes, Allen, "*Representations of Spectral Coordinates in FITS*", 2006, http://arxiv.org/abs/astro-ph/0507293

[19] Rots et. al., "*Representations of Time Coordinates in FITS*", recent draft, http://hea-www.cfa.harvard.edu/~arots/TimeWCS/WCSPaperV0.981Letter.pdf

[20] R, Plante (ed.), A. Stébé, K. Benson, P. Dowler, M. Graham, G. Greene, P. Harrison, G. Lemson, A. Linde, G. Rixon & IVOA Registry-WG, *VODataService: a VOResource Schema Extension for Describing Collections and Services Version 1.1.,* IVOA Recommendation, 02 December 2010. http://www.ivoa.net/Documents/VODataService/1.1

[21] P. Dowler, G. Rixon, D. Tody, DAL-WG, *Table Access Protocol Version 1.0,* IVOA Recommendation 27 March 2010. http://www.ivoa.net/Documents/TAP/1.0

[22] A. H. Rots (ed.), *Space-Time Coordinate Metadata for the Virtual Observatory Version 1.33*, IVOA Recommendation 30 October 2007. http://www.ivoa.net/Documents/STC/1.33

[23] R. J. Hanisch, *Resource Metadata for the Virtual Observatory*, Version 1.12, IVOA Recommendation 02 March 2007. http://www.ivoa.net/documents/latest/RM.html

[24] R. J. Hanisch, C. Arviset, F. Genova, B. Rhino, *IVOA Document Standards,* IVOA Recommendation 13 April 2010. http://www.ivoa.net/Documents/DocStd/