International
Virtual
Observatory
Alliance

The Observation Core Components Data Model
Version 1.0-20100520

IVOA Working Draft May 22, 2010

This version:
http://www.ivoa.net/Documents/Notes/WD-ObsCoreDM-0.2-20100520.pdf

Latest version:
http://www.ivoa.net/Documents/latest/ObsCoreComponentsDM.html

Previous versions:
0.1

Editor(s):
Mireille Louys, François Bonnarel

Authors:
Mireille Louys, François Bonnarel, Alberto Micol,
David Schade, Anita Richards, Patrick Dowler
Jesus Salgado, Doug Tody,
Igor Chilingarian, Daniel Durand

Abstract

This document discusses the definition of the core components of the Observation data model that are necessary to cover data discovery use-cases when querying data centers. It exposes the use-cases to be carried out, explains the model and provides a table of fields to be implemented in a Table Access Protocol (TAP) TAP service. Such a small model is easy to understand and implement by data providers that wish to publish their data into the Virtual

Observatory.

Status of this document

This document has been produced by the Data Model Working Group. It organises the metadata attached to an astronomical observation and develops an abstract view in the Observation data model Core Components part. On the other hand it provides a full guideline to implement such a model in the Table Access Protocol framework.

Due to the DM and DAL aspects of this document, this will be reviewed by both Working Groups.

Acknowledgements

This work has been partly funded by Euro-VO AIDA project that we acknowledge here. CADC implementations ??? ESO implementation prog?

Contents

1 Introduction

Modeling of observational metadata has been a long term activity in the IVOA since it was created in 2002. Various modeling efforts like Resource Metadata, STC, Spectrum data model, and Characterisation data model, have been recommended and are currently used in IVOA services and applications. This work now reaches a mature state where we need to homogeneise the various description and access ways to discover, retrieve and/or analyse any kind of data products resulting from an astronomical observation. However data model implementation among data providers is still poorly represented except for the most common data model fields used to publish a data collection in the Registry (based on the Resource metadata DM) or in protocols like the well supported SSA or SIA. This situation is also reinforced by more technical problems: serialisation (pure XML or Utypes), and protocols for metadata access were not always available for practical implementation of these data models. The emergence of the TAP protocol allows now to use a generic method for all kind of datasets and models. Definition of TAP services implementing the Core of Observation data model for various archives will provide a unified discovery interface to a large set of heterogeneous data archives of images, cubes, spectra and catalogs. The goal of the current note is to propose a subset of Utypes of the overall Observation data model currently under construction that enables to support a full set of use-cases collected from the community, and to show how it can be implemented as a TAP service. Such a simple and general service should be easily implemented by any data provider, for any kind of data.

The document is composed as following: Section 2 presents the use cases collected within the astronomical community by the IVOA Uptake committee. Section 3 identifies useful features for our purpose in the Observation/Characterization data model and defines the ObsCore data model. Section 4 specifies the required data model fields as they are used in the TAP service: table names, column names, column datatype, utype from the Observation Core components data model, and required units. Section 5: TBD. Section 6 is the conclusion. Appendix A contains the Use-cases gathered from the community and listing various possible search criteria in the data discovery process. ??? Appendix B provides examples of Obs/TAP queries and describes implemented services.

2 Use cases

We first focused on data discovery use-cases, aimed at finding observations in the VO domain, by broadcasting the same query to a bunch of data centers or to all VO subscribers.

Ultimately we need to provide data providers with a list of item and features that they could easily map to their database system, in order to answer to the defined queries.

The goal is to be simple enough to be implementable, and not to be exhaustive on all exotic data sets.

The main features of these use-cases are mainly: - multi-wavelength search - multi-types of data (spectrum, cube , catalogs) Refined or advanced searches may include extra knowledge stemming from astronomical objects classification and would need to extract results from catalogs, possibly by using fine sub-queries.

Here we list just one example in each use-case category but the full list is available in Appendix B or at http://....

2.1 Discover imaging data of interest

Use-Case 1.2: Let me input a list of RA and DEC coordinates and show me spatially coincident data that satisfies

1.2.1 Data type is Imaging or spectroscopy data
1.2.2 Includes one or more of the RA,DEC
1.2.3 Includes both a wavelength in the range 5000-900 angstroms AND an X-ray image (AND=SERVREQ)

2.2 Discover Spectral data of interest

Use-case 2.2: Show me a list of all data that satisfies

2.2.1 DataType=Spectrum
2.2.2 Wavelength includes 6500 angstroms
2.2.3 Spectral Resolution better than 15 angstroms
2.2.4 Spatial Resolution better than 2 arcseconds FWHM
2.2.5 Exposure Time ¿ 3600 seconds
2.2.6 Data Quality = Any

2.3 Discover Data cubes of interest

Use-Case 3.4 : Show me a list of all data that satisfies

3.4.1 DataType=cube
3.4.2 RA includes 16.00
3.4.3 Dec includes +41.00
3.4.4 Wavelength includes 6500 angstroms
3.4.5 Wavelength includes 4000 angstroms
3.4.6 Spectral resolution better than 5 angstroms
3.4.7 Exposure time more than 3600 seconds
3.4.8 Data Quality= Fully Calibrated

2.4 Discover general data of interest

Use-Case 5.3 : Show me a list of all data that satisfies

5.3.1 DataType=Imaging or Spectroscopy
5.3.2 RA includes 16.00 hours
5.3.3 DEC includes +41.00 degrees
5.3.4 SDSS images and spectra AND CFHTLS images and spectra

A common denominator to most of the use-cases defined here is the set of physical features in terms of spatial, spectral, temporal, and photometric properties? This is well covered by the Characterisation Data Model and can be re-used as a package within the Observation DM. The Characterisation DM as well as the Spectrum DM define lists of Utypes [?] that are ready to use for our cases. Access and identification metadata, defined and summarized in the SSA Utype list [?], provide the necessary mechanism to identify and retrieve a data set.

2.5 How to answer these use cases?

The idea is to build up a VO interface to allow data providers to simply describe their observation metadata stored in DBMS systems and provide a query mechanism for users to discover, then retrieve the data products. The currently developed TAP protocol offers the appropriate implementation layer. Here we specify the data model and the relational model that TAP services must implement. The strength of this approach is that users will be able to submit the identical query to all TAP services that implement the ObsCore model and get back uniform results. This can be described in a standard way in the `TAP_SCHEMA!.`Queries to this database may be done in all cases by ADQL as well as other query languages supported by individual services (for example using PQL when that standard is finalised and implemented). The generation of Utypes from a data model is not described here.

We assume it will be consistent with the Utype definition WD (Louys et al, 2009).

3 The Observation Core Components Data Model

Let us just sketch out how the core components fit into the scope of the general Observation Data Model and list which classes and attributes will be used to support the above use-cases.

3.1 The context and history of metadata modeling in the VO

The Observation data model project appeared at the first Data Model forum held at the May 2003 IVOA meeting in Cambridge,UK. Rapidly some main classes appeared to be necessary to organize the metadata: Dataset or Observation, Identification, Physical Characterisation, Provenance (either instrumental or software) and Curation. A description of the early stages of this development can be found in [**?**],(Observation IVOA note). In parallel an effort dedicated to spectra was lead by the DM Working group. The Spectrum data model represents all necessary metadata for one specific type of observational data: simple spectra. For the overall Observation Data Model, the physical characterization has been identified to be on first priority already in 2004. It was completed as an IVOA recommendation after 4 years of discussion which included computer scientists, astronomers and data providers under the lead of J.Mc Dowell.

3.1.1 Characterisation data model

The Characterisation data model organizes metadata as a 3D matrix spanning independently the various physical axes (spatial, spectral, time, flux or whatever observable quantity), four levels of granularity, and some features or Properties (coverage, resolution and sampling). This scheme allows to support selection of data sets for data discovery as well as data analysis.

3.1.2 New efforts

While the Characterisation data model was setting up a logical framework to describe the properties/features of each observation in the VO, the approach here is more pragmatic and leads to a simple implementation data model, and its protocol application using the emerging TAP/SCHEMA framework. In the mean time, the Generic Data set/Observation data model is currently developed integrating Curation and DataID description borrowed from the Spectrum Data model with a detailed description of the Provenance (instrumental and computational) for observed datasets. The consistency between the two efforts is a major goal, and will be warranted by defining the Core components to be re-usable at the more general level.

3.2 Observation data model Core Components and attributes definition

To fulfill our goal of serialization of the ObsCore data model in TAP we have to select the most useful subset of utypes necessary for our use cases. Most of them do actually belong to characterization with the exception of the contentType, the target name, and the collection name.

From all the metadata covered and described in the full Observation data model, only a small part is needed to support data discovery in a regular and efficient manner. We then concentrate first on the definition of core components of the Observation data model that will be used to support the use-cases described above. The Observation Core component data model is summarized in a UML class diagram below; this is the logical data model. The implementation of such a model is described in Section 4.

Here we list all the metadata used as selection criteria in the listed use-cases and in the response data center might give back.

3.2.1 UML description of the model

The data model for observation is organised according to some Object Oriented Programming principles in order to define unique and consistent concepts , as re-usable classes. UML helps to sketch out the class organisation as shown in Fig. TODO. 1. This class diagram covers all classes used in the context of the Observation Core components Model. The Characterisation classes, describing how the data span along the main physical axes, are used here partly, and only the attributes relevant to this modeling effort are shown here. This is also the case for the DataID and Curation classes extracted from the Spectrum/SSA data model where only a subset of attributes are necessary for data discovery.We consider for now that we use Characterisation classes only down to the Support level.

Encoding the coordinates attributes depends on the nature of each Characterisation axis and will be described in detail in the Full Observation DM. The Utypes shown on both tables provide the inner structure of each class attribute. Therefore we do not develop the SpatialAxis, TimeAxis classes on the diagram , for the sake of clarity.

3.2.2 Type of Observation

The model defines a *data product type* attribute for the Observation Class. It is the type of observation the user queries for or selects for retrieval. This is coded as a string that conveys a general idea of the content and organisation of a dataset. We consider a coarse classification of the types of dataset interesting for science usage, covering images, cubes, spectra, light curves, SED, etc... The Observation.DataProductType attribute takes its string value in the following set, organized according to up to 4 levels of granularity:

- **Image**
  - Image.2DSkyImage
  - Image.2D any 2D image: weight maps, bad-pixel map, etc: TBD
  - Image.Longslit 2D image for a long slit spectrum with one axis mapped on wavelength
  - Image.Cube 2D+ extra dimension(s)
  - Image.Cube.Spectral
  - Image.Cube.Time
  - Image.Cube.Polarization
- **Spectrum**
  - Spectrum.1D
  - Spectrum.SED
  - Spectrum.Polarisation (TO BE DISCUSSED)...
  - Spectrum.Echelle
  - Spectrum.IFU (TBD)
- **TimeSeries**
  - TimeSeries
  - TimeSeries.LightCurve flux variable with Time
  - TimeSeries.RadialVelocity (TBD)
- **Visibility**
  - Visibility.Image
  - Visibility.Cube
- **EventList**

This provides a hierarchy of possible data product types that is stated here but can be extended by data providers in the future depending on new kinds of data and search procedures.. We give the possibility here for data provider to extend this classification and return an extend data product type. However, this should start with some element given in the hierarchy above and should not be redundant with any already existing string defined above.

3.2.3 Calibration level

It is a convention we suggest to use to classify the different possible calibration status of an observed dataset. This is up to the data provider to consider how to map his own internal classification to the suggested scale here.

Following examples can help to find the most appropriate value for the *calibLevel* attribute.

- Level 0:
  raw instrumental data, possibly in proprietary internal provider format, that need specific tools to be handled.
- Level 1:
  Instrumental data in a standard format(FITS, VOTable, SDFITS, ASDM, etc. The data may or may not be calibrated. Standards tools can handle it.
- Level 2:
  Science ready data , with instrument signature removed, and calibration status defined on all physical axes.
- Level 3:
  Enhanced data products like mosaics, improved co-added image cubes, resampled or drizzled images, etc. spectra with calibrated velocity axis at a particular line rest frequency. In such case, the improved calibration procedure is described by the data provider in some way, progenitors of such a data product can be identified into the reduction pipeline.

This classification is simple enough to cover all regimes. Data providers will adjust the mapping of their various internal levels of calibration to this general frame, with the knowledge of the PIs for each project.

4 Implementation of ObsCore in a TAP Service

The ObsCore model will be implemented within Table Access Protocol (TAP) services such that all valid queries can be executed unchanged on any service that implements the model. Here we specify an explicit mapping of the model to relational database tables; in the context of TAP this means we are specifying the logical tables as described in the `TAP_SCHEMA` and VOSI-tables metadata content. This does not necessarily imply that the underlying database will have the identical structure, but in most cases the relationship between `TAP_SCHEMA` description and the underlying tables is straightforward.

`schema_name`	`table_name`	description
`ivoa`	`ivoa.ObsCore`	ObsCore 1.0

Table 1: TAP underscore SCHEMA.tables description of the ObsCore model

Tables 1 and 2 provides the primary information needed to describe the ObsCore model in terms of `TAP_SCHEMA` tables and columns. The ”constraint” specified in 2 above is not part of the the `TAOP_SCHEMA.columns` description, but is required by the ObsCore model and specified here to make this clear to implementors. Additional standard content for the individual columns is specified below. See Appendix C for a usable SQL script that inserts the standard `TAP_SCHEMA` content.

4.1 Data Product Type

- `column_name`: `dataproduct_type`
- datatype: adql:VARCHAR
- size: implementor decides
- units: NULL
- utype: Observation.DataProductType
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `dataproduct_type` column contains a simple string value describing the primary nature of the data product. Allowed values are: image, spectrum, timeseries, visibility, eventlist, cube. Values are in lower case.

table_name	column_name	datatype	units	constraint
ivoa.ObsCore	dataproduct_type	adql:VARCHAR		not null
ivoa.ObsCore	calib_level	adql:INTEGER		not null
ivoa.ObsCore	obs_collection	adql:VARCHAR		not null
ivoa.ObsCore	obs_id	adql:VARCHAR		not null
ivoa.ObsCore	obs_publisher_did	adql:CLOB		not null
ivoa.ObsCore	access_url	adql:CLOB		
ivoa.ObsCore	access_format	adql:VARCHAR		
ivoa.ObsCore	access_estsize	adql:INTEGER	KB	
ivoa.ObsCore	obs_target	adql:VARCHAR		
ivoa.ObsCore	s_ra	adql:DOUBLE	deg	
ivoa.ObsCore	s_dec	adql:DOUBLE	deg	
ivoa.ObsCore	s_fov	adql:DOUBLE	deg	
ivoa.ObsCore	s_region	adql:REGION	deg	
ivoa.ObsCore	s_resolution	adql:DOUBLE	arcsec	
ivoa.ObsCore	t_min	adql:DOUBLE	d	
ivoa.ObsCore	t_max	adql:DOUBLE	d	
ivoa.ObsCore	t_exptime	adql:DOUBLE	sec	
ivoa.ObsCore	t_resolution	adql:DOUBLE	sec	
ivoa.ObsCore	em_min	adql:DOUBLE	m	
ivoa.ObsCore	em_max	adql:DOUBLE	m	
ivoa.ObsCore	em_res_power	adql:DOUBLE		
ivoa.ObsCore	o_fluxucd	adql:VARCHAR		

Table 2: TAP underscore SCHEMA.columns description of the ivoa.ObsCore table

Subtypes, separated by one or more dots from the base categorisation, may be specified in provider-specific columns (see 4.20).

Values in the `dataproduct_type` column must not be NULL.

TODO: reconsider extending to two-level product type when the data model in previous section is completed.

4.2 Calibration Level

- `column_name`: `calib_level`
- datatype: adql:INTEGER
- size: NULL
- units: NULL
- utype: Observation.calibLevel
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `calib_level` column tells the user the amount of calibration processing that has been applied to create the data product. Allowed values are: 0 (instrumental/raw data in a non-standard format), 1 (raw data in a standard format), 2 (calibrated data in standard format, instrument signature removed), and 3 (more highly processed data product). Data providers decide which value best described their data products.

Values in the `calib_level` column must not be NULL.

4.3 Collection Name

- `column_name`: `obs_collection`
- datatype: adql:VARCHAR
- size: implementor decides
- units: NULL
- utype: Observation.DataID.collection
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `obs_collection` column identifies the data collection to which the data product belongs. A data collection can be any collection of datasets which are alike in some fashion. Typical data collections might be all the data from a particular telescope, instrument, or survey. The value is either the registered shortname for the data collection, the full registered IVOA identifier for the collection, or a data provider defined shortname for the collection.

Values in the `obs_collection` column must not be NULL.

4.4 Observation Identifier

- `column_name`: `obs_id`
- datatype: adql:VARCHAR
- size: implementor decides
- units: NULL
- utype: Observation.DataID.collectionID?
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `obs_id` column contains a collection-specific identifier for an observation. In the case where multiple data products are available for an observation (e.g. with different calibration levels), the `obs_id` value will be the same for each product of the observation.

Values in the `obs_id` column must not be NULL.

TODO: Determine correct utype.

4.5 Publisher Dataset Identier

- `column_name`: `obs_publisher_did`
- datatype: adql:CLOB
- size: implementor decides
- units: NULL
- utype: Observation.DataID.PublisherDID
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `obs_publisher_did` column contains the IVOA dataset identifier [ref] for the published data product. This value is globally unique since different publishers (data centres) that provide access to a data product will each assign their own value.

We specify the datatype as CLOB in the TAP service so that users will know they can only use the `obs_publisher_did` column in the select clause of a query. That is, users cannot specify this column as part of a condition in the where clause and implementors are free to generate the URL during output.

Values in the `obs_publisher_did` column must not be NULL.

4.6 Access URL

- `column_name`: `access_url`
- datatype: adql:CLOB
- size: NULL
- units: NULL
- utype: Observation.Access.reference
- UCD: ?
- principal: 1
- indexed: 0
- std: 1

The `access_url` column contains a URL that can be used to download the data product (files).

We specify the datatype as CLOB in the TAP service so that users will know they can only use the `access_url` column in the select clause of a query. That is, users cannot specify this column as part of a condition in the where clause and implementors are free to generate the URL during output.

4.7 Access Format

- `column_name`: `access_format`
- datatype: adql:VARCHAR
- size: NULL
- units: NULL
- utype: Observation.Access.format?
- UCD: ?
- principal: 1
- indexed: 0
- std: 1

The `access_format` column contains a string indicating the format of downloaded the data product (files). The values are made up of dot-deparated words that describe (in increasing detail) the structure of the data file(s).

Allowed values include any legal mimetype [REF] that describes the format or one of the following short forms: fits, fits.image, fits.image.mef, fits.table, fits.cube, fits.cube.mef, directory, votable, csv, tsv. TODO: Doug and Arnold to provide list by 2010-06-15.

TODO: Determine correct utype.

4.8 Estimated Download Size

- `column_name`: `access_estsize`
- datatype: adql:BIGINT
- size: NULL
- units: KB
- utype: Observation.Access.size?
- UCD: ?
- principal: 1
- indexed: 0
- std: 1

The `access_estsize` column contains an approximate size (in KB) of the file(s) available via the `access_url`.

TODO: Determine correct utype.

4.9 Target Name

- `column_name`: target
- datatype: adql:VARCHAR
- size: implementor decides
- units: NULL
- utype: Observation.Target.name
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `target` column contains the name of the target of the observation. This is typically the proper name of an astronomical object, but could be the name of a survey field.

4.10 Central Coordinates

- `column_name`: `s_ra`

- datatype: adql:DOUBLE
- size: NULL
- units: deg
- utype: Characterisation.SpatialAxis.Coverage.Location.coord.Position2D.Value2.C1
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `s_ra` column stores the ICRS Right Ascension of the centre of the observation.

- `column_name`: `s_dec`
- datatype: adql:DOUBLE
- size: NULL
- units: deg
- utype: Characterisation.SpatialAxis.Coverage.Location.coord.Position2D.Value2.C2
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `s_dec` column stores the ICRS Declination of the centre of the observation.

4.11 Spatial Extent

- `column_name`: `s_fov`
- datatype: adql:DOUBLE
- size: NULL
- units: deg
- utype: Characterisation.SpatialAxis.Coverage.Extent?
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `s_fov` column contains the approximate size of the region covered by the data product. For a circular region, this is the diameter (not the radius).

4.12 Spatial Coverage

- `column_name`: `s_region`
- datatype: adql:REGION
- size: NULL
- units: deg
- utype: Characterisation.SpatialAxis.Coverage.Support
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

We specify the datatype as the logical type REGION so that users can specify spatial queries using a single column and in a limited number of ways. If included in the select list of the query, the output is always an SCT-S string as described in [TAP, section 6]. In the where clause, the `s_region` column can be used with the ADQL geometry functions (INTERSECTS, CONTAINS) to specify conditions; the service will generally have to translate these into native SQL that enforces the same conditions or a suitable approximation. Implementors may approximate the spatial query conditions by translating the INTERSECTS and CONTAINS function calls in the query.

In addition, ADQL specifies several functions which may take the `s_region` column as an argument: AREA, CENTROID, and COORDSYS. The AREA function returns the area (in sq. deg.) of the observed the region. In cases where the `s_region` itself is an approximation (a bounding box, for example), this function should still return the actual value. This may be implemented by computing and storing the area in a separate column and converting the AREA(`s_region`) function call into a column reference in the query. The CENTROID function returns an ADQL POINT value; if used in the select list the output is always an STC-S string as described in [TAP, section 6]. The coordinates must be the same as those found in the `s_ra` and `s_dec` columns, which are provided for convenience. The COORDSYS function returns the coodrinate system used for the `s_region`; in the ObsCore model this is restricted to ICRS, so this can be implemented by converting the COORDSYS(`s_region`) function call to a constant in the query.

4.13 Spatial Resolution

- `column_name`: `s_resolution`
- datatype: adql:DOUBLE

- size: NULL
- units: arcsec
- utype: Characterisation.SpatialAxis.Resolution.refVal.Cresolution
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

TODO: define here...

4.14 Time Bounds

- `column_name`: `t_min`
- datatype: adql:DOUBLE
- size: NULL
- units: d
- utype: Characterisation.TemporalAxis.Coverage.Bounds.limits.Interval.StartTime
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `t_min` column contains the start time of the observation in Modified Julian Day(s).

- `column_name`: `t_max`
- datatype: adql:DOUBLE
- size: NULL
- units: d
- utype: Characterisation.TemporalAxis.Coverage.Bounds.limits.Interval.StopTime
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `t_min` column contains the stop time of the observation in Modified Julian Day(s).

4.15 Exposure Time

- `column_name`: `t_exptime`
- datatype: adql:DOUBLE

- size: NULL
- units: sec
- utype: Characterisation.TemporalAxis.Coverage.Support.Extent
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `t_exptime` column contains the exposure time. For simple exposures, this is just `t_max - t_min` expressed in seconds. For data where the detector is not active at all times (e.g. products made by combining exposures taken at different times), the `t_exptime` will be smaller than `t_max - t_min`. For data where the `t_exptime` is not constant over the enture data product, the median exposure time per pixel is a good way to characterise the typical value. In all cases, `t_exptime` is generally used as an indicator or relative sensitivity (depth) within a single collection (e.g. `obs_collection`); providers should supply a suitable relative value when it is not feasible to define or compute the true exposure time.

4.16 Time Resolution

- `column_name`: `t_resolution`
- datatype: adql:DOUBLE
- size: NULL
- units: sec
- utype: Characterisation.TemporalAxis.Resolution.refVal
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

TODO: define here...

4.17 Energy Bounds

- `column_name`: `em_min`
- datatype: adql:DOUBLE
- size: NULL
- units: m
- utype: Characterisation.SpectralAxis.Coverage.Bounds.limits.Interval.LoLim
- UCD: ?

- principal: 1
- indexed: implementor decides
- std: 1

The `em_min` column contains the minimum energy observed, expressed as (Barycentric? vacuum?) wavelength.

- `column_name`: `em_max`
- datatype: adql:DOUBLE
- size: NULL
- units: m
- utype: Characterisation.SpectralAxis.Coverage.Bounds.limits.Interval.HiLim
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `em_max` column contains the maximum energy observed, expressed as (Barycentric? vacuum?) wavelength.

4.18 Spectral Resolution

- `column_name`: `em_res_power`
- datatype: adql:DOUBLE
- size: NULL
- units: NULL
- utype: Characterisation.SpectralAxis.Resolution.ResPower.refVal
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `em_res_power` column contains the typical or characteristic resolving power of the energy axis. The value is dimensionless (e.g. delta lambda / lambda).

4.19 Observable Axis Description

- `column_name`: `o_fluxucd`
- datatype: adql:VARCHAR
- size: implementor decides
- units: NULL

- utype: Characterisation.ObservableAxis.ucd
- UCD: ?
- principal: 1
- indexed: implementor decides
- std: 1

The `o_fluxucd` column contains a UCD [ref] describing the values of the observable within the data product. The observable is the measured quantity, for example the pixel value in an image.

TODO: Doug and Arnold to provide a list of observables, Francois and Mireille to map thsi to a list of typical UCDs by 2010-06-15.

4.20 Additional Columns

Service providers may include additional columns in the ivoa.ObsCore table to expose additional metadata. These columns must be described in the `TAP_SCHEMA.columns` table and in the output from the VOSI-tables resource. Users may access these columns by examining the column metadata for individual services and then using them explicitly in queries or by selecting all columns in the query (e.g. "select * from ivoa.ObsCore ..." in an ADQL query).

5 Registering a TAP Service with the ObsCore Model

TAP services that implement the ObsCore model should be registered to indicate this fact so that users can easily find all services that accept ObsCore queries. Initially, this can be done by using the keyword "ObsCore" to describe the service. Fione-grained registries may include the complete VODataService `tableset` description, but this is not available through all searchable registries. The TAP extension schema [REF] allows service providers to specify which IVOA data models are used within a TAP service. Services that implement the ObsCore model should use the registry extension schema mechanism to publicise their support with:

```
standardID="ivo://ivoa.net/std/ObsCore/v1.0"
```

placed (somewhere) in the TAP Registry extension schema [REF], where `ivo://ivoa.net/std/ObsCore/v1.0` is the IVOA identifier of the VOStandard [REF] defined in this document.

References

Appendix A: Data discovery Use-cases

Use-cases to be included here see original document.

Appendix B: Example of ADQL queries for some of our use cases.

Here we consider a very general use case from the list (1.1) :
Show me a list of all data that satisfies:

1.1.1 Datatype=any
1.1.2 contains RA=16.0 and DEC=40.0

These data would be searched on all VO services by sending the following query:

```
SELECT * FROM ivoa.Obscore WHERE
s_ra_min < 16.0 AND s_ra_max > 16.0 AND
s_dec_min < 40.0 AND s_dec_max > 40.0
```

More constraints can be added in the following use-case (1.3):
Show me a list of all data that satisfies

1.3.1 DataType=Image
1.3.2 Spatial resolution better than 0.3 arcseconds
1.3.3 Filter = J or H or K
1.3.4 RA between 16 hours and 17 hours
1.3.5 DEC between 10 degrees and 11 degrees

```
SELECT * FROM ivoa.Obscore_ext WHERE
dataproduct_type=‘‘Image.2D’’
AND  s_resolution < 0.3
AND s_ra > 240 AND s_ra < 255 AND
s_dec > 10 AND s_dec < 11
AND
(em_min > 2.1 AND em_max < 2.4) OR
(em_min >= 1.6 AND em_max <= 1.8) OR
(em_min >= 1.2 AND em_max <= 1.4)
```

Appendix C: Sample SQL

All TAP services that implement the ObsCore model will have the same table and column descriptions in their `TAP_SCHEMA`, with the exception of the service-specific column descriptions as described in 4. Specifically, implementations may change the description, the indexed flag, and the size (in the case of variable width columns).

Note: The following SQL was modified but not tested to match the requirements in section 4. A tested version will be included as soon as possible.

```
delete from TAP_SCHEMA.columns where table_name = 'ivoa.ObsCore';

delete from TAP_SCHEMA.tables where table_name = 'ivoa.ObsCore';

delete from TAP_SCHEMA.schemas where schema_name = 'ivoa';

insert into TAP_SCHEMA.schemas (schema_name,description) values
( 'ivoa',
  'collection of tables for IVOA data models with standard relational mappings')

insert into TAP_SCHEMA.tables (schema_name,table_name,utype,description) values
( 'ivoa', 'ivoa.ObsCore', 'Observation', 'ObsCore 1.0' );

-- NOTE: values in the size column are implementation-specific
--       and should be set by providers
-- NOTE: values in the description may be changed/enhanced by providers
-- NOTE: all other columns have values mandated by the specification
insert into TAP_SCHEMA.columns
(table_name,column_name,datatype,unit,utype,ucd,size,principal,indexed,std,descr
values
( 'ivoa.ObsCore', 'dataproduct_type', 'adql:VARCHAR', NULL,
  'Observation.DataProductType', NULL, 128, 1,0,1,
  'type of data product (image, spectrum, timeseries, visility, eventlist, cube)

( 'ivoa.ObsCore', 'calib_level', 'adql:VARCHAR', NULL,
  'Observation.calibLevel', NULL, NULL, 1,0,1,
  'calibration level (0,1,2,3)' ),

( 'ivoa.ObsCore', 'target', 'adql:VARCHAR', NULL,
  'Observation.Target.name', NULL, 128, 1,0,1,
  'name of the intended target' ),
( 'ivoa.ObsCore', 'obs_collection', 'adql:VARCHAR', NULL,
  'Observation.DataID.collection', NULL, 128, 1,0,1,
  'name of the archive or data collection' ),
( 'ivoa.ObsCore', 'obs_id', 'adql:VARCHAR', NULL,
  'TBD:Observation.DataID.collectionID', NULL, 128, 1,0,1,
  'IVOA dataset identifier assigned by the publisher' ),
( 'ivoa.ObsCore', 'obs_publisher_did', 'adql:CLOB', NULL,
  'Observation.Curation.PublisherDID', NULL, NULL, 128, 1,0,1,
  'IVOA dataset identifier assigned by teh publisher' ),

( 'ivoa.ObsCore', 'access_url', 'adql:CLOB', NULL,
  'Observation.Access.reference', NULL, NULL, 1,0,1,
  'URL to download the data' ),
( 'ivoa.ObsCore', 'access_format', 'adql:VARCHAR', NULL,
  'TBD:Observation.Access.format', NULL, 128, 1,0,1,
  'file format or structure of downloaded product(s)' ),
( 'ivoa.ObsCore', 'access_estsize', 'adql:INTEGER', 'KB',
  'TBD:Observation.Access.size', NULL, NULL, 1,0,1,
  'estimated download size' ),

( 'ivoa.ObsCore', 's_ra', 'adql:DOUBLE', 'deg',
  'Characterisation.SpatialAxis.Coverage.Location.coord.Position2D.Value2.C1',
  NULL, NULL, 1,0,1,
  'ICRS Right Ascension of central coordinates' ),
( 'ivoa.ObsCore', 's_dec', 'adql:DOUBLE', 'deg',
  'Characterisation.SpatialAxis.Coverage.Location.coord.Position2D.Value2.C2',
  NULL, NULL, 1,0,1,
  'ICRS Declination of central coordinates' ),
( 'ivoa.ObsCore', 's_region', 'adql:REGION', 'deg',
  'Characterisation.SpatialAxis.Coverage.Support',
  NULL, NULL, 1,0,1,
  'region bounded by observation' ),
( 'ivoa.ObsCore', 's_resolution', 'adql:DOUBLE', 'arcsec',
  'Characterisation.SpatialAxis.Resolution.refVal.Cresolution',
  NULL, NULL, 1,0,1,
  'typical/representative/charateristic spatial resolution' ),

( 'ivoa.ObsCore', 't_min', 'adql:DOUBLE', 'd',
  'Characterisation.TemporalAxis.Coverage.Bounds.limits.Interval.StartTime',
  NULL, NULL, 1,0,1,
  'start time of observation (MJD)' ),
( 'ivoa.ObsCore', 't_max', 'adql:DOUBLE', 'd',
  'Characterisation.TemporalAxis.Coverage.Bounds.limits.Interval.StopTime',
  NULL, NULL, 1,0,1,
  'end time of observation (MJD)' ),
( 'ivoa.ObsCore', 't_exptime', 'adql:DOUBLE', 'sec',
  'Characterisation.TemporalAxis.Coverage.Support.Extent',
  NULL, NULL, 1,0,1,
  'exposure time of observation' ),
( 'ivoa.ObsCore', 't_resolution', 'adql:DOUBLE', 'sec',
  'Characterisation.TemporalAxis.Resolution.refVal',
  NULL, NULL, 1,0,1,
  'typical/representative/charateristic temporal resolution' ),

( 'ivoa.ObsCore', 'em_min', 'adql:DOUBLE', 'm',
  'Characterisation.SpectralAxis.Coverage.Bounds.limits.Interval.LoLim',
  NULL, NULL, 1,0,1,
  'minimum wavelength' ),
( 'ivoa.ObsCore', 'em_max', 'adql:DOUBLE', 'd',
  'Characterisation.SpectralAxis.Coverage.Bounds.limits.Interval.HiLim',
  NULL, NULL, 1,0,1,
  'maxmimum wavelength' ),
( 'ivoa.ObsCore', 'em_res_power', 'adql:DOUBLE', NULL,
  'Characterisation.SpectralAxis.Resolution.ResPower.refVal',
  NULL, NULL, 1,0,1,
  'typical/representative/charateristic spectral resolution' ),

( 'ivoa.ObsCore', 'o_fluxucd', 'adql:VARCHAR', NULL,
  'Characterisation.ObservableAxis.ucd', NULL, 32, 1,0,1,
  'UCD describing the observable axis (pixel values)' )
;
```

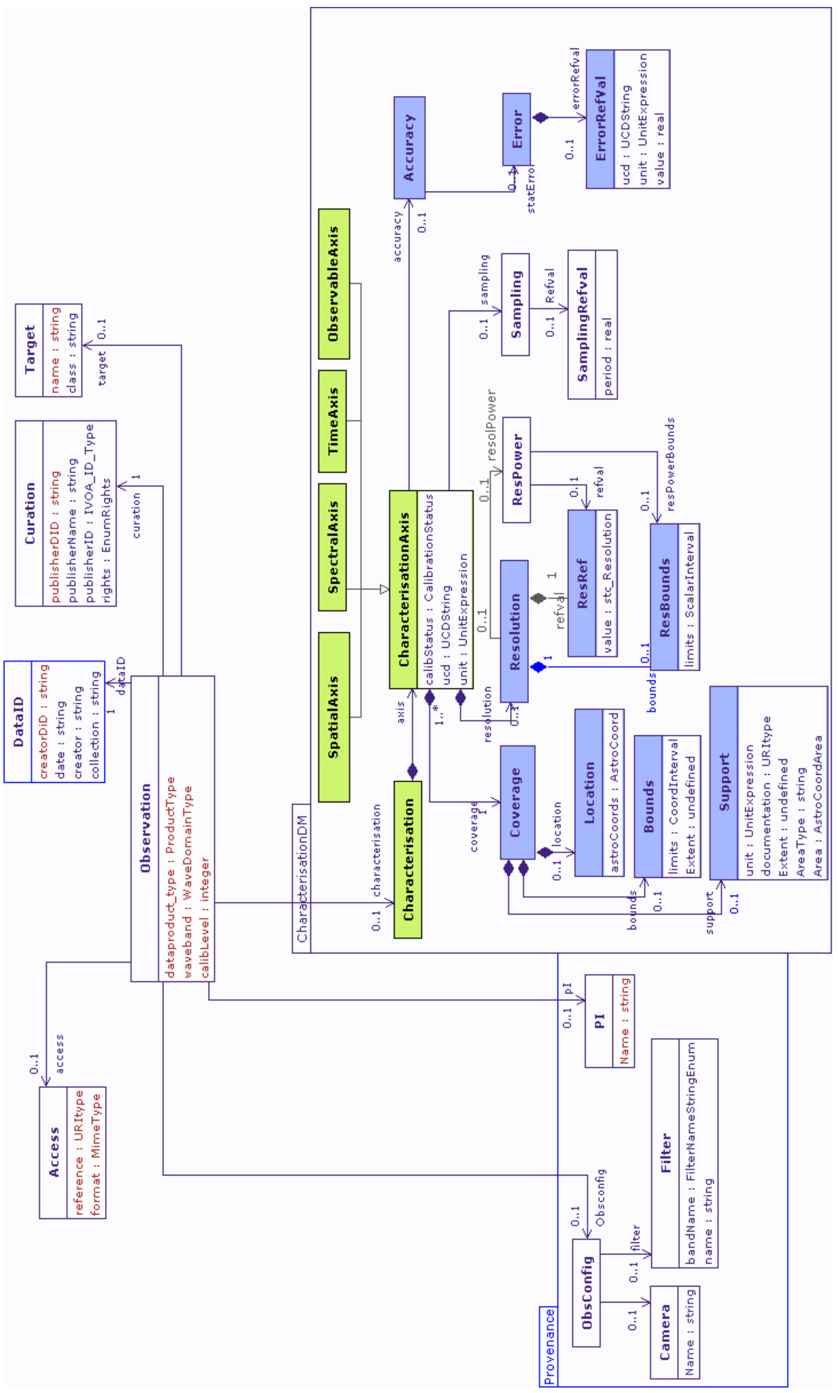

Figure 1: *Here is the class diagram representing the classes used to organise observational metadata. Classes may be linked together via an association or aggregation link. The minimal set of necessary attributes for data discovery is shown in brown.*