Time Series Cube Data Model

Version 1.1

IVOA Note 2017-12-15

Abstract

The purpose of this document is to create a data model and its serialization for expressing generic time series data. Already existing IVOA data models are reused as much as possible. The model is also made as generic as possible to be open to new extensions but at the same time closed for modifications. This enables maintaining interoperability throughout different versions of the data model.

We define the necessary building blocks for metadata discovery, serialization of time series data and understanding it by clients.

We present several categories of time series science cases with examples of implementation. We also take into account the most pressing topics for time series providers like tracking original images for every individual point of a light curve or time-derived axes like frequency for gravitational wave analysis.

The main motivation for the creation of a new model is to provide a unified time series data publishing standard - not only for light curves but also more generic time series data, e.g., radial velocity curves, power spectra, hardness ratio, provenance linkage, etc.

The flexibility is the most crucial part of our model - we are not dependent on any physical domain or frame models. While images or spectra are already stable and standardized products, the time series related domains are still not completely evolved and new ones will likely emerge in near future. That is why we need to keep models like *Time Series Cube DM* independent of any underlying physical models. In our opinion, this is the only correct and sustainable way for future development of IVOA standards.

Status of This Document

This is an IVOA Note expressing suggestions from and opinions of the authors. It is intended to share best practices, possible approaches, or other perspectives on interoperability with the Virtual Observatory. It should not be referenced or otherwise interpreted as a standard specification.

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

Contents

Acknowledgements

1 Term Definition

- Data cube - When we are speaking about a data cube, we have in mind the pure database point of view. In astronomy a term data cube is often implicitly understood as an image cube or a spectral cube and we do not want to be restricted only to these domains. In this document, a data cube is much closer to an OLAP data cube (Kimball (1997*a*)) (while again not restricting us to business-related domains). This data cube is usually implemented in the database as a star-schema with its dimensions being our axis domains (Kimball (1997*b*)).

- Time Series Cube DM - The *Time Series Cube data model (DM)* defines data structures for holding metadata and data of a time series data cube.

- Axis domain - By axis domain we mean the physical interpretation of an axis of the data cube. This is usually axis metadata, e.g., spectral limits of that axis, resolution, etc. From the IT perspective, while *Time Series Cube DM* holds the *data*, axis domain model stores the relationships and context which enables us to get *information* from the data.

- Dataset - We mean an IVOA Dataset as specified in *Observation Data Model* (Mireille Louys (2011)), also known as the *ObsCore DM*, i.e. a collection in which can be stored any kind of observation. As such, a dataset is holding information common for every element of its collection, e.g., facility, instrument or creator.

2 Introduction

This is a working draft of a new approach for representing generic time series data. It was designed to be modular and extensible, new changes will extend the model, not modify it. Extending a model means reusing what was in the extended model, while adding new structures to it. That said, any part of this design is open to discussion. If you have any ideas how it can be improved, please contact us on Time Domain Interest Group mailing list voevent@ivoa.net.

We can incorporate any kind of axis in the data cube, where all other up-to-now approaches tried to embed domain-specific metadata inside the model. They described specialized axes relevant only for light curves - SimpleTimeSeries (Matthew Graham (2014)) and SSAP-based approaches (Doug Tody (2012)). For these approaches, making any enhancement to the model required re-work of what was already in the model, not an extension.

The description of axis domain, i.e. the metadata which enables us to extract information and knowledge from the data, is kept outside of this model. The relationships towards other models can be found in Chap. 3.

Compared to previous approaches, we are only referencing the physical domain specific metadata, instead of incorporating them inside the *Time Series Cube DM*. The detailed description of the *Time Series Cube DM* can be found in Chap. 4

The position of *Time Series Cube DM* inside IVOA Architecture (Christophe Arviset (2010)) can be seen on Fig. 1.
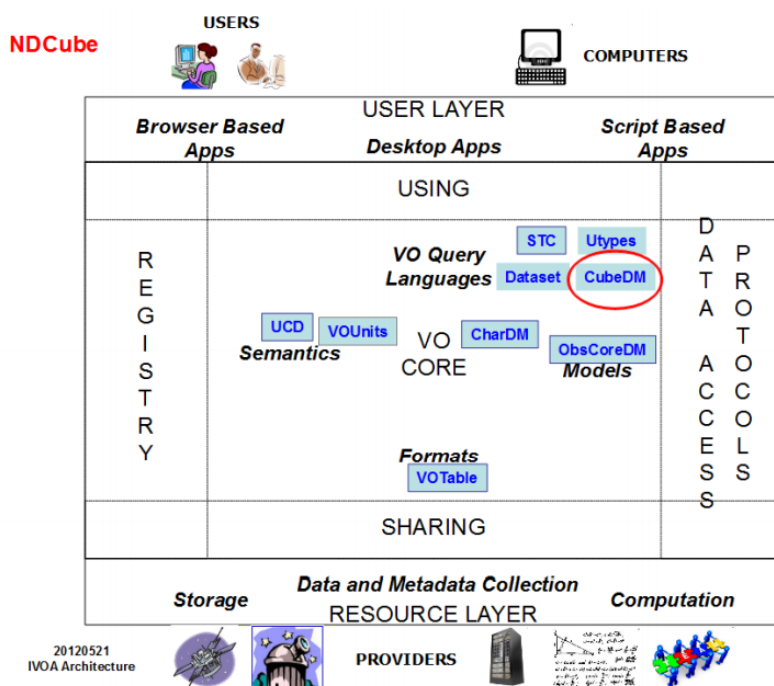
Figure 1: IVOA architecture

3   Dependent data models

In this chapter we describe *Time Series Cube DM* position and relationships to other models.

On Fig. 2 we can see other models that we import without modifications, i.e. copy them inside our model, and without extending them.

On Fig. 3 we show the *Time Series Cube DM* relationships to other data cube models.
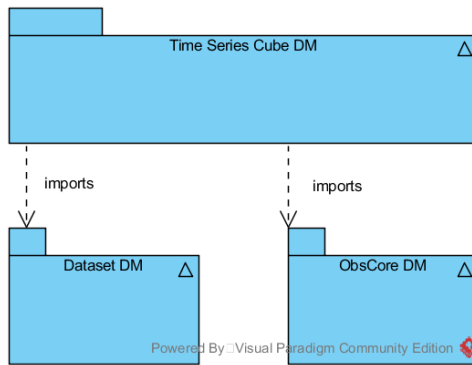
Figure 2: This UML describes models that we import AS-IS, without any modifications to them or extending them.
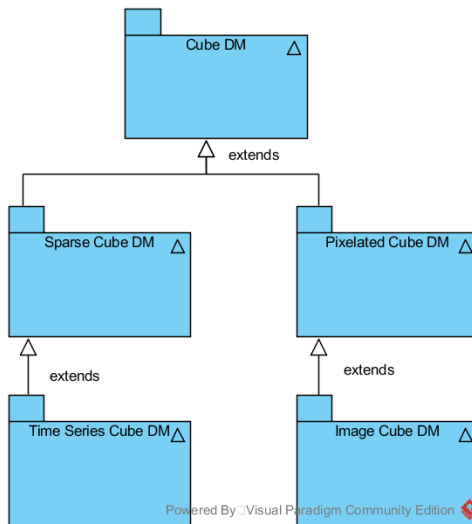
Figure 3: UML model specifying the position of the *Time Series Cube DM* compared to the *Cube DM* and the *ImageCube DM*.

3.1   Imported Data models

We import these data models without modification. We basically copy the model definition into *Time Series Cube DM*, making us completely dependent on what is defined within these - *Time Series Cube DM* wouldn't work without them.

However, because we use them as black boxes, it does not matter if these data models change - *Time Series Cube DM* has no common parts shared with them.

3.1.1 Dataset DM

The *Dataset DM* is used for describing common information for the whole set of time series cubes. The individual cubes, i.e. elements of the dataset collection described by *Dataset DM*, are then described within the *Time Series Cube DM*.

3.1.2 ObsCore DM

ObsCore DM in combination with TAP (Mireille Louys (2011)) is used for discovery of the time series cube metadata and we use it without modification. The *ObsCore DM* has already support for time series data with `dataProductType=timeseries` parameter. We are now working with a reference implementation, using GAVO DaCHS for server side and modified Splat-VO (Petr Skoda (2014)) for consumption of this ObsCore TAP service and discovering time series. More about discovery can be found in Chap. 5.

3.1.3 VO-DML

The *Time Series Cube DM* uses the VO-DML (Gerard Lemson (2016)) for annotating parts of the model and referencing individual axis domain models from inside the time series cube. The serialization of *Time Series Cube DM* model into VO-DML is not the purpose of this document.

We use VO-DML because it is the only reasonable way how to reference an actual model, not only entities already defined by that model. This is useful for referencing axis domains from within the axis definitions inside the *Time Series Cube DM*.

We are not importing the whole VO-DML, the only purpose it serves us is brining a simple way to reference other models as a whole, not only their structures.

3.2 Extended Data models

The following models are either base classes or siblings with relationships to the *Time Series Cube DM*. We reuse parts of them or reference them and in some cases, we override what has been defined in them if it was not generic enough for the purposes of *Time Series Cube DM*.

3.2.1 Cube DM

The *Cube DM* describes generic data cubes and is defined in the *N-Dimensional Cube/Image Model* (et al. (2015)) document. This document is also describing the *Sparse Cube DM* and the *Pixelated Cube DM*, however, it is not describing *Image Cube DM* (as could be erroneously understood from the title). We will refer to this standard as *N-Dimensional Cube DM*.

3.2.2 Pixelated Cube DM

This data model describes an abstraction of pixelated data cubes, e.g., image cubes. Essentially this is still the same data cube as the generic one, we only put restrictions on the main axes. The pixelated cube must have pixelated axes, i.e., regular intervals between coordinates.

3.2.3 Image DM

The *Image Cube DM* is currently being worked upon in the *IVOA Image DM* (Douglas Tody (2013)) working draft but is not being aligned completely with the *Cube DM* effort. It sis an extension of *Pixelated Cube DM* and the image axes have naturally regular intervals between their coordinates.

3.2.4 Sparse Cube DM

This data model extends the *Cube DM* while adding restrictions on the main axes. The main axes must hold sparse data, i.e., we can have irregular intervals between the coordinates.

3.2.5 Time Series Cube DM

The *Time Series Cube DM* is describing time series data cubes, which are by definition sparse. We describe them in more detail in Chap. 4.

4 Time Series Cube DM

In this chapter we work with the time series data cubes defined within the *Time Series Cube DM*. These data cubes are instances of the `Time Series Cube` class seen on Fig. 4.

Note that the purpose of the *Time Series Cube DM* metadata is to enlist all the axes of the cube and provide linkage towards their values and errors, not to completely describe axis domains.

Also, after intensive discussions, we decided to move statistical metadata about the axis of our data cubes to be moved into a really small external model *Quantity DM* because this will be useful not only for cubes but any numerical data in general.

The axis domain descriptions will be held within other specialized data models, e.g., spatial and time domain in STC (Rots (2007)), spectral domain in Photometry DM (Jesus Salgado (2013)), etc. These external models will only be referenced by the *Time Series Cube DM*. This is where we differ from the original *Cube DM* that already focuses on supporting SIAv2 (Patrick Dowler (2015a)) image-based protocol and we do not want to restrict ourselves only to pixelated types of data cubes.

Withdrawing axis domain definitions as a dependency from the *Time Series Cube DM* helps us in the future when new data models will be created, e.g., for probability distributions or gravitational wave domains. If such a new data model appears, we can just add it as a reference to axis metadata within the data cube and the *Time Series Cube DM* does not need to be changed at all. It also solves the main drawback of previous approaches and that is extensibility - SSAP or SimpleTimeSeries standards would need to be changed if a new domain was to be supported. Moreover, such a domain would not be logically part of spectral or time aspects of those models and therefore it does not belong there.

The only difference between axis coordinate and the value for that coordinate is that the axis coordinates are independent while the data values are fully functionally dependent on these coordinates.

Therefore, we can afford (from the data point of view) to treat them equally, which is one of the biggest advantages of a data cube. We can browse the data cube by filtering the axis coordinates and the data values in the same way. This enables a client for example to slice the time axis of a light curve (coordinates) within an interval and the magnitude axis (values) within another interval using the same algorithm.

The time series data cube serialized accordingly to the *Time Series Cube DM* describes only the data, not the physical interpretation. From the IT perspective, it holds *data*, not *information* or *knowledge*. We are building on the fact that any information that we would like to store within the time series data cube, even the provenance, can be represented by some value and can be treated as an axis from the data point of view.

The `Time Series Cube` class as seen on Fig. 4 is the main class of *Time Series Cube DM*. It extends the `Sparse Cube` class taken from *N-Dimensional Cube DM*. This means that we are reusing what was already defined within `Sparse Cube` class coming from *Sparse Cube DM* and add new parameters in the `Time Series Cube` class.

The UML representation of our model can be seen on Fig. 4. On this figure we describe two data models, i.e., *Dataset DM* and *Cube DM*. The *Axis Domain DMs* package is collecting external models that we reference. *Time Series Cube DM* imports classes from the *DataSet DM* model.

A time series data cube can bee distinguished from a more generic sparse data cube by having the `dataProductType` attribute with value `"timeseries"`.

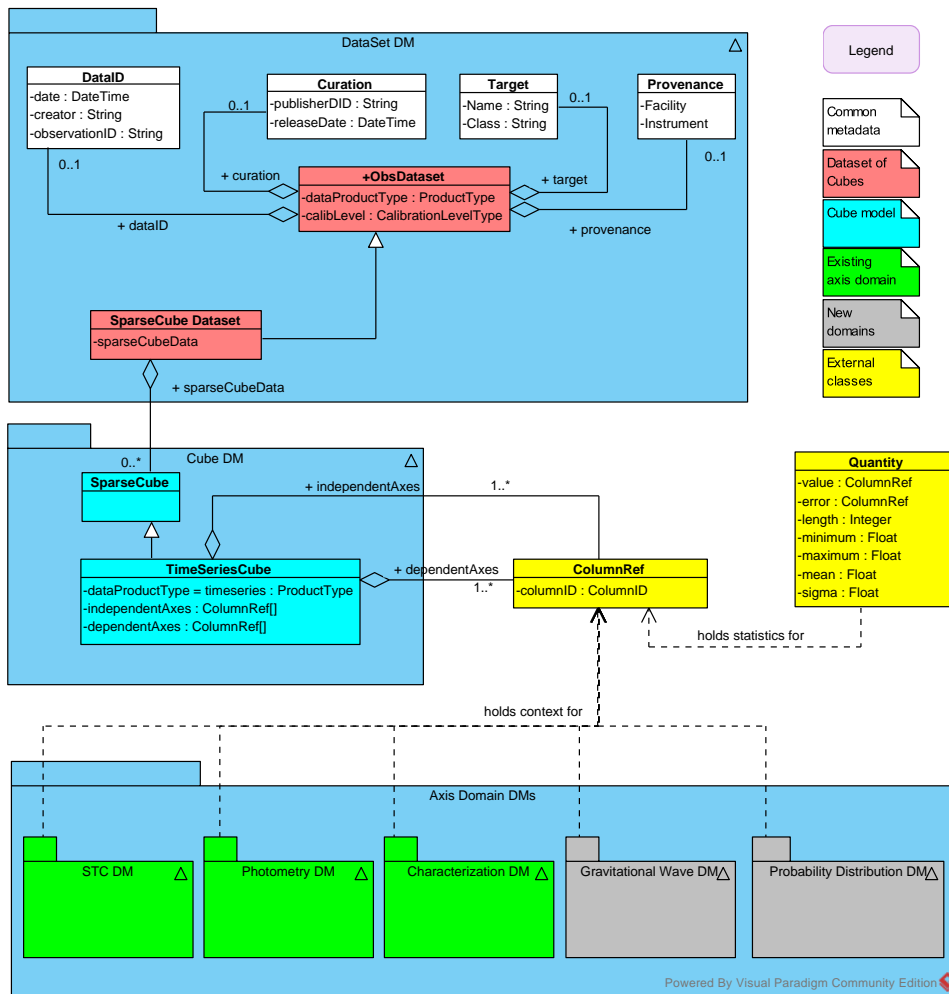The *DataSet DM*, *Cube DM* and *Axis Domain* packages seen on Fig. 4 are described below.

Figure 4: The UML diagram of Time Series Cube DM ecosystem. The *DataSet DM* classes (red) collect sparse cubes and time series cubes (cyan). The `Axis Domain DMs` referenced by the `CubeAxis` class are split between already existing (green) and potentially coming (grey) models.

11

4.1 DataSet DM

Everything you see in the *Dataset DM* package on Fig. 4 is imported, i.e., not changed within the scope of this document.

The *DataSet DM* holds common metadata for all `Time Series Cube` class instances collected within this dataset and stores them in the `ObsDataset` class. Typically that would be `DataID` class attributes like `creator`, `Curation` class metadata like `publisherDID` or `releaseDate` and `Provenance` class metadata like `facility` or `instrument` identifying the origin of the data. A `Target` class can also have its metadata specified, if the dataset has a common observation target.

The *SparseCube Dataset* class extends *ObsDataset* class and puts restriction on what can be stored within this collection. An instance of the *SparseCube Dataset* class has an attribute `sparseCubeData` which holds the collection of *SparseCube* class instances, i.e. `Time Series Cube` instances.

4.2 Cube DM

In this section we speak about relevant classes from the *Cube DM* package, i.e., `Sparse Cube` class and `Time Series Cube` class as described on Fig. 4.

4.2.1 Sparse Cube class

On Fig. 4 we show the `Sparse Cube` class that we are re-using from *N-Dimensional Cube DM*. The `Time Series Cube` class is extended from this one.

4.2.2 Time Series Cube class

A client can identify the `Time Series Cube` instance by having the keyword `timeseries` in the `dataProductType` attribute of its instance. By using only *ColumnRef* class which is already referenced from *Axis Domain models*, we override the *Cube DM* dependency on axis domains, e.g., *STC v2.0 DM*.

A client reading the data cube might want to know what should it treat as independent (coordinates) and dependent (data) values - they are collected within the `independent_axes` and in `dependent_axes` attributes of the `Time Series Cube` instance.

4.2.3 ColumnRef class

`independent_axes` and in `dependent_axes` inside `Time Series Cube class` use the `ColumnRef` which is a primitive element of the serialization used. In VOTable this would be `FieldRef` element, in FITS table just the `name` of the column.

If a specialized client wants to find more about specific metadata and data of an axis (its frame, domain, all related parameters and fields), it will find this information where he is used to it - in metadata of the dataset and in the parameters of the column itself. In VOTable this would be special `Param` elements at the beginning of the table and attributes of `Field` element, e.g., `name`, `type`, `unit`, `ucd`, etc.

And because this client is specialized towards that domain, it understands metadata in the referenced domain-specific data model (e.g., gravitational wave domain, spectral domain, etc.) and we don't need to make it part of *Time Series Cube DM*.

4.2.4 Quantity class

This class was originally part of the *Cube DM* but we moved it out because statistical information is useful for images and spectra, basically any kind of numerical data. We are still mentioning it here though because we believe that it will have bigger impact on *Cube DM* than other types of data.

Its main role is to describe statistical distribution of data within the axes of our data cube allowing us to make more informed decisions about what subset of the data we would like to filter. Knowing mean and sigma values or even quantiles of the data and filtering it based on these (e.g. "I want to filter only points that are 3-5 sigma from the mean value.") will provide much better results than trying to guess the ranges based only on minimum and maximum value of that axis.

The Quantity class is referencing the *ColumnRef* class as it must say which column (also error column) it is describing in the dataset. Making it dependent only on this primitive type is again in accordance with the loose coupling principle we are using - making the higher level models independent of each other.

4.3 Axis Domain DMs

An answer to what all can be described with an axis (the axis domain) is not within the scope of this document. If we want to describe our spectral coordinates (e.g., list of filters) by the *Photometry DM* or in *STC DM* or add a completely new model for my axis domain (e.g., Gravitational Wave data), it does not affect the *Time Series Cube DM*.

These models, however, do always reference a primitive type of their serialization, such as `ColumnRef`. By using this primitive type dependency (loose coupling for all of our data models), we are essentially making *Cube DM*, *Quantity DM* and *Axis Domain DMs* independent of each other.

4.4   Time Series Cube DM summary

With the growing amount of models and their diversity within the IVOA, keeping models independent of each other is the only viable approach to their evolution and maintenance. By only referencing a primitive type instead of an external model or even importing every detail of that model, the *Time Series Cube DM* does not need to change every time an *Axis Domain Model* changes or a new one is created - the abstraction of that domain will be still the same. This information is valid also for the *Quantity DM*.

5   Discovery - ObsCore

The ObsCore protocol is the natural option for discovery of our time series data cubes. In ObsCore 1.1, the most crucial parameters are added for the description of axis lengths for the basic types of axes, i.e., `s_xel` for spatial axis length, `t_xel` for temporal, etc., so the client knows what is the size of the cube it is about to download. The usual implementation of ObsCore model discovery is using TAP protocol for access to the tables holding the time series data.

At current point of time, the ObsCore suffices our requirements for discovery and anyone can add new parameters to his ObsCore service - except the *DataLink* (Patrick Dowler (2015*b*)) metadata discovery. This point is under discussion and, in the end, it may will not demand change to the ObsCore TAP standard.

6   Serialization

We decided to discuss the VOTable format as the first one to support as it is the easiest one to form discussions upon and afterwards close them with a deterministic XSD that can validate whether the serialization is strictly following the standard or not. This section focuses on serialization of the metadata and data defined within the *Time Series Cube DM*, not on serialization of the data model structure - that will be handled by VO-DML.

`GROUP` XML element can be used as an equivalent of the data model classes described in Chap. 4. These are linked to the IVOA data models by annotations. The *Time Series Cube DM* itself then contains `independent_axes` and `dependent_axes GROUP` elements which hold the actual information about the axis.

The metadata, i.e., common information for every point in that axis, is referenced in the group as `PARAMRef` elements and the data (actual measurements that have usually different values for every point of this axis) is referenced by `FIELDRef` elements.

These `FIELDRef` elements point to `FIELD` elements, holding data of the VOTable. The order of these is the same as the order of data values stored in the standard `DATA` element of the VOTable (here we can find TR and TD elements of the VOTable).

Axis Models are again serialized as `GROUP` elements and are referenced from the independent or dependent axis collections by the model using `GROUPRef`.

The basic example of how such a serialization would look like can be seen on Fig. 5 and more can be found in the Chap. 7.

Please note that we enhanced and completed the serialization examples for version 1.1 and updated them in 7.

7 Supported science cases

In this chapter we describe how the most typical time series use cases are supported. We show that our approach can support all of the usual use cases, such as simple light curves or groups of light curves, in the same way as completely generic time series data like provenance linkage or frequency-based analysis.

We focus mainly on the support of time series data other than light curves. We will describe the science cases collected and suggest a VOTable serialization example for each one of these examples, so we can see that the *Time Series Cube DM* is easily serializable.

7.1 Requirements

The formalization of basic time series use cases comes from the *CSPTime-series* science cases document (Solano (2012)). In addition to these use cases, we collected a few others from LSST or LIGO requirements for time series data models. In this chapter, we take these science cases one by one and explain to implement them with *Time Series Cube DM* .

We took the first three logical groups of light curves from the *CSPTi-meseries* document (Solano (2012)) and added two more, the complex light curves and gravitational waves, as examples of what can be accomplished with the model.

7.2 Science case definitions

The science case definitions are defined from the requirements and are as follows.

- Simple Light curves - Combine photometry and light curves of a given object/list of objects in the same photometric band. We refer to the

Group A in the *CSPTimeseries* document mentioned above as *simple light curves.*

- Groups of light curves - Combine photometry and light curves of a given object/list of objects in different photometric bands. We refer to the Group B in the *CSPTimeseries* document and slightly generalize it as *groups of light curves*, i.e., not only collecting light curves for filters, but also holding other types of axes, e.g., polarization.

- Generic Time series - Time series other than light curves. Basically anything what has at least one time-based axis can be put here. We refer to the Group C in the *CSPTimeseries* document as *generic time series.*

- Complex light curves - Special requirements on light curve data:

  - Provenance tracking: These use cases require to track original image or its cutouts for every point of the light curve, e.g., link to SIAP cutouts of the relevant region in the original image which can be opened in a graphical client. We are able to transfer the data and describe the metadata for this use case.

  - Probability distribution: Another use case computes and transfers probability distribution functions for the time series, e.g., computed simple Gaussian distribution parameters for the coordinate errors from which the light curve was generated). We are able to transfer the data but for metadata description, there would have to be created a new kind of Probabilistic DM as seen on Fig. 4. However, when such model comes, we just add the `model` reference to the *CubeAxis* metadata, without any changes to the *Time Series Cube DM* serialization.

  - *DataLink* for time series: A basic use case for *Time Series Cube DM* combined with *DataLink* would be storing data linkage for the whole time series instance, e.g., *DataLink* to the periodogram of this time series. Another one would be filtering of light curve points on a supported criteria, e.g., filtering only high quality points of a light curve. Here we are struggling with the metadata transfer for the ObsCore TAP discovery described in Chap. 5. The problem is that with *ObsCore DM* we can get also other data than time series and the *DataLink* does not support metadata restriction based on a `dataProductType`. We are able to construct such a *DataLink* for the light curves, the only problem is with transferring the metadata about such a domain over ObsCore TAP service.

- Power spectra, Gravitational waves: Special use case for generic time series data - we show here how to store not only simple time axis

16

but also its derivatives, e.g., frequency. For Gravitational waves, this would be tracking for example ASD (strain per rtHz) against Frequency (Hz) as seen on the Gravitational wave tutorial (*Signal Processing with GW150914* (n.d.)).

7.3 Simple Light curves

This is the basic use case supported up to now with *SimpleTimeSeries* (Matthew Graham (2014)). Now, we can easily support it in the *Time Series Cube DM*. An example of the *DataSet* metadata can be seen on Fig. 5, the definition of the cube on Fig. 6 and 7 and the first row on Fig. 8.

The light curve rendered within SPLAT-VO client ((Petr Skoda (2014))) can be seen on Fig. 9 and 11. These images were taken on real data coming within our reference implementation for server and client. More about the reference implementation can be found in Chap. 8.

7.3.1 Examples

The examples of simple light curves can be seen on Fig. 5, Fig. 6, Fig. 7 and Fig. 8. The plot rendered within SPLAT-VO can be seen on Fig. 9.

Figure 5: Dataset metadata serialization via VO-DML. There is a dataset (collection) of sparse cubes. The dataset contains the metadata, e.g., the `observationID` or `creator` metadata. All of the serialized metadata are annotated against their respective counterparts in *Dataset DM* and *Time Series Cube DM* using the `dmRole` attribute.

7.4 Groups of light curves

The use case of storing light curves for one object in multiple photometric bands can be supported simply by addition of one independent axis - the

17

```
43              <INSTANCE ID="ndgnsoliggsa" dmtype="ndcube:Cube">
44                  <ATTRIBUTE dmrole="independentAxis">
45                      <COLUMN ref="HJD"/>
46                  </ATTRIBUTE>
47                  <ATTRIBUTE dmrole="dependentAxis">
48                      <COLUMN ref="raj2000"/>
49                      <COLUMN ref="dej2000"/>
50                      <COLUMN ref="MAG"/>
51                      <COLUMN ref="ABMAG"/>
52                      <COLUMN ref="STMAG"/>
53                      <COLUMN ref="PHOTRATE"/>
54                      <COLUMN ref="PHOTON"/>
55                      <COLUMN ref="FLX"/>
56                      <COLUMN ref="FNU"/>
57                      <COLUMN ref="FLAM"/>
58                  </ATTRIBUTE>
59              </INSTANCE>
```

Figure 6: Time series data cube metadata serialization. Here we can see the `ndcube:Cube dmtype` along with the axis definitions. The independent axes like temporal or spatial are referencing the `FIELD` elements holding their data. It works the same way for both independent and dependent axes.

```
62              <INSTANCE ID="ndgnsolommha" dmtype="q:Quantity">
63                  <ATTRIBUTE dmrole="length">
64                      <CONSTANT ref="ndgnsoelnwea"/>
65                  </ATTRIBUTE>
66                  <ATTRIBUTE dmrole="minimum">
67                      <CONSTANT ref="ndgnsooapgpa"/>
68                  </ATTRIBUTE>
69                  <ATTRIBUTE dmrole="maximum">
70                      <CONSTANT ref="ndgnsoelnasa"/>
71                  </ATTRIBUTE>
72                  <ATTRIBUTE dmrole="value">
73                      <COLUMN ref="raj2000"/>
74                  </ATTRIBUTE>
75                  <ATTRIBUTE dmrole="mean">
76                      <CONSTANT ref="ndgnsooapdpa"/>
77                  </ATTRIBUTE>
78              </INSTANCE>
79              <INSTANCE ID="ndgnsolomhsa" dmtype="q:Quantity">
80                  <ATTRIBUTE dmrole="length">
81                      <CONSTANT ref="ndgnsoelnwea"/>
82                  </ATTRIBUTE>
83                  <ATTRIBUTE dmrole="minimum">
84                      <CONSTANT ref="ndgnsoelneea"/>
85                  </ATTRIBUTE>
86                  <ATTRIBUTE dmrole="maximum">
87                      <CONSTANT ref="ndgnsoelnsha"/>
88                  </ATTRIBUTE>
89                  <ATTRIBUTE dmrole="value">
90                      <COLUMN ref="dej2000"/>
91                  </ATTRIBUTE>
92                  <ATTRIBUTE dmrole="mean">
93                      <CONSTANT ref="ndgnsoelnupa"/>
94                  </ATTRIBUTE>
95              </INSTANCE>
```

Figure 7: Statistical information about the data cube. In here we can see the `q:Quantity dmtype`. The statistical information is held within the `constant` elements and the reference to the column that is actually holding the data can be recognized by `dmrole="value"` attribute. Note that for describing statistical information we don't need to have the context held in axis domain metadata.

spectral axis. Having the light curve points in multiple bands, e.g., UBVRI, is nothing more than having one more discrete spectral axis with coordinates that can have 5 distinct values. If the spectral axis is described in continuous intervals, the error `FIELD` would be meaningful, for just a list of filter names we are storing only data value elements, i.e. `value` attribute of `Quantity`,

```xml
<FIELD ID="HJD" datatype="double" name="HJD" ucd="VOX:Image_MJDateObs" unit="d" utype="spec:Spectrum.Data.SpectralAxis.Value">
    <DESCRIPTION>Epoch at midpoint of observation in Heliocentric Modified Julian Date</DESCRIPTION>
</FIELD>
<FIELD ID="raj2000" datatype="double" name="raj2000" ucd="pos.eq.ra;meta.main" unit="deg">
    <DESCRIPTION>Observed RA of the object</DESCRIPTION>
</FIELD>
<FIELD ID="dej2000" datatype="double" name="dej2000" ucd="pos.eq.dec;meta.main" unit="deg">
    <DESCRIPTION>Main value of declination</DESCRIPTION>
</FIELD>
<FIELD ID="FLX" datatype="float" name="FLX" ucd="phot.flux">
    <DESCRIPTION>Photon flux</DESCRIPTION>
</FIELD>
<FIELD ID="FLXERR" datatype="float" name="FLXERR" ucd="stat.error;phot.flux"/>
<FIELD ID="MAG" datatype="float" name="MAG" ucd="phot.mag" unit="mag">
    <DESCRIPTION>Magnitude of the object</DESCRIPTION>
</FIELD>
<FIELD ID="MAGERR" datatype="float" name="MAGERR" ucd="stat.error;phot.mag" unit="mag"/>

<DATA> <!-- the order of data colums are always the same as order of FIELD elements defined above. -->
    <TABLEDATA>
        <TR>
            <TD>56287.11921047</TD>
            <TD>12.9768538030848</TD>
            <TD>-72.9580158132637</TD>
            <TD>7.86796e-17</TD>
            <TD>0.0</TD>
            <TD>19.0829</TD>
            <TD>0.0</TD>
        </TR>
    </TABLEDATA>
</DATA>
```

Figure 8: Time series data cube data serialization. Here we can see the FIELD elements referenced from the time series cube and quantity metadata. This is a basic light curve holding the heliocentric Julian date for temporal axis, equatorial coordinates and two dependent axes with measurements - flux and magnitude.
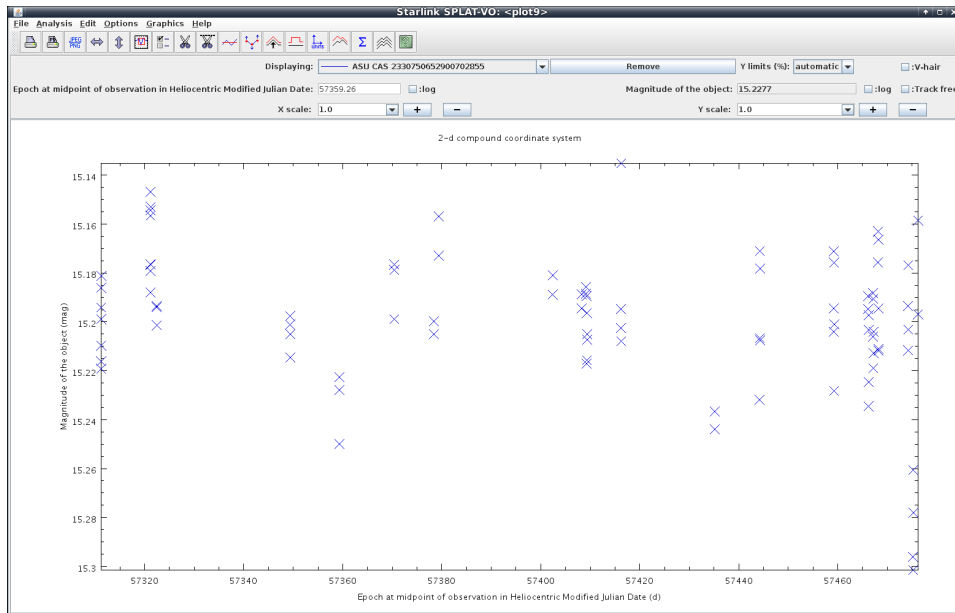
Figure 9: Light curve within SPLAT-VO. The light curve is rendered from VOTable coming within time series data cube format. Note that SPLAT-VO is able to distinguish the magnitude and inverts the axis, as well as the units for the temporal axis. These are coming from the axis domain metadata.

with textual information.

It is evident that adding more such "groups" (other independent axes) will produce a lot of duplicated values in the DATA element of our VOTable.

However, this is just a question of optimization.

If the need arises, for start, compressing algorithms such as EXI (*Efficient XML Interchange (EXI) Format, Version 1.0* (2014)) can be used for immediate value of eliminating the duplicities. A cleaner approach would be a table normalization during the transfer - the independent axes (coordinates) would be stored in separate `TABLE` elements (the same goes for FITS) that would be referenced by the dependent axes records. This, however, would need an extension to VOTable standard to allow referencing not only between elements but also within data. The requirement is that a value in one data column must be able to reference another value within another table.

The optimization of serialization is not the main aim of this document but can be expanded in the future if deemed necessary.

7.4.1 Examples

The XML example for groups of light curves can be found on Fig. 10. The plot rendered within SPLAT-VO can be seen on Fig. 11.

```
43              <INSTANCE ID="ndgnsoliggsa" dmtype="ndcube:Cube">
44                  <ATTRIBUTE dmrole="independentAxis">
45                      <COLUMN ref="HJD"/>
46                      <COLUMN ref="band"/>
47                  </ATTRIBUTE>
48                  <ATTRIBUTE dmrole="dependentAxis">
49                      <COLUMN ref="raj2000"/>
50                      <COLUMN ref="dej2000"/>
51                      <COLUMN ref="MAG"/>
52                      <COLUMN ref="ABMAG"/>
53                      <COLUMN ref="STMAG"/>
54                      <COLUMN ref="PHOTRATE"/>
55                      <COLUMN ref="PHOTON"/>
56                      <COLUMN ref="FLX"/>
57                      <COLUMN ref="FNU"/>
58                      <COLUMN ref="FLAM"/>
59                  </ATTRIBUTE>
60              </INSTANCE>
```

Figure 10: Groups of light curves time series data cube metadata serialization. The *Dataset* and *Quantity* information are the same, the only change we need to do is to add one more axis to the independent collection. This means that the dependent axis (actual measurements, i.e., magnitude, flux) can hold values for any combination of the date and spectral axis values.

7.5 Generic time series

The Generic time series are the main motivation for using *Time Series Cube DM*, as any kind of data that can be structured into series can be represented by a data cube. In here we also show that we can describe our metadata by using `PARAM` elements which our client can understand by parsing `UCDs`. Or, we can describe those by using external data models that the *Time Series Cube DM* only references.
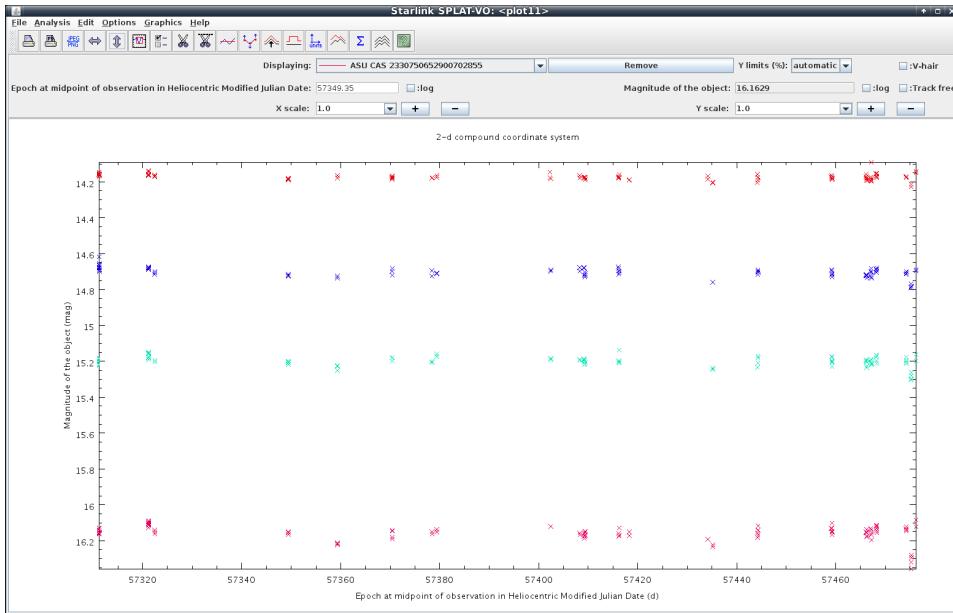
Figure 11: Group of light curves within SPLAT-VO. As we are still using a valid VOTable serialization, our SPLAT-VO client can render the time series data cube for a light curve taken in 4 filters without modification. The spectral axis is the 3rd rendered axis and is expressed in colors instead of axis coordinates. The axis labels are taken from axis domain metadata already modeled within Photometry DM

The fact that the *Time Series Cube DM* will not know about the details of every domain metadata is crucial as it enables us to describe the metadata with our own parameters if a standard model is not yet available, without the need of including them in *Time Series Cube DM*. After the model is standardized, e.g., *STC v2.0, Provenance DM, Probability DM*, etc., we just need to add new stuff defined within that model without the need to change anything within the serialization defined by *Quantity DM* or *Cube DM*.

The axes derived from the basic types can be still tracked as the same type. The frequency for a power spectrum is still a time series in its own sense and we can easily support it within *Time Series Cube DM*. It is true that the the frequency is not technically a sparse axis that we chose as a basis for the time series data cubes, but logically a power spectrum or periodogram can be understood as time series. Moreover, the *Sparse Cube DM* places a restriction on *Time Series Cube DM* to support sparse axes, not to restrict itself only to them.

Another option for supporting these cases would be simply to move our generic definitions from *Time Series Cube DM* to *Cube DM* and not worry about whether a periodogram is a time series when we already can agree

that i definitely can be described as a data cube.

7.5.1 Examples

A basic example of a generic time series can be seen on Fig. 12. Focus on the highlighted model attribute of the hardness ratio dependent axis. Even if the axis domain model does not exist yet, we still want to describe the structure of the data (values and errors of this axis) and store the rest in `PARAM` elements, which are not part of the data-perspective view provided by *Time Series Cube DM*. When the axis domain model for hardness ration arrives, we just add new elements from it to the VOTable without the need to change any existing, maintaining full backwards-compatibility.

```
1   <!--================= Cube annotation =========================-->
2
3   <INSTANCE ID="ndgnsoliggsa" dmtype="ndcube:Cube">
4       <ATTRIBUTE dmrole="independentAxis">
5           <COLUMN ref="HJD"/>
6       </ATTRIBUTE>
7       <ATTRIBUTE dmrole="dependentAxis">
8           <COLUMN ref="hardnessRatio"/>
9       </ATTRIBUTE>
10  </INSTANCE>
11
12  <!--================= Quantity annotation =========================-->
13
14  <INSTANCE ID="ndgnsobbbasa" dmtype="q:Quantity">
15      <ATTRIBUTE dmrole="value">
16          <COLUMN ref="hardnessRatio"/>
17      </ATTRIBUTE>
18      <ATTRIBUTE dmrole="error">
19          <COLUMN ref="hardnessRatioError"/>
20      </ATTRIBUTE>
21  </INSTANCE>
22
23  <!--================= VOTable columns =========================-->
24
25  <FIELD ID="HJD" datatype="double" name="HJD" ucd="VOX:Image_MJDateObs" unit="d" >
26      <DESCRIPTION>Epoch at midpoint of observation in Heliocentric Modified Julian Date</DESCRIPTION>
27  </FIELD>
28  <FIELD ID="hardnessRatio" datatype="float" name="hardnessRatio" >
29      <DESCRIPTION>Hardness ratio</DESCRIPTION>
30  </FIELD>
31  <FIELD ID="hardnessRatioError" datatype="float" name="hardnessRatio" >
32      <DESCRIPTION>Hardness ratio statistical error</DESCRIPTION>
33  </FIELD>
34
35  <DATA>
36      <TABLEDATA>
37          <TR>
38              <TD>56287.11921047</TD>
39              <TD>1.0</TD>
40              <TD>0.05</TD>
41          </TR>
42      </TABLEDATA>
43  </DATA>
44
```

Figure 12: Hardness ratio time series example. The data cube enlists the axes. The errors for these axes can be found in the *Quantity* metadata (along with the statistics, which we omit here for simplicity) and the column definitions which are storing the actual data. When a model for hardness ratio arrives, it would certainly specify ucd for the hardnessRatio `FIELD` element.

7.6 Complex light curves

This group is already describing science cases gathered apart the ones described in *CSPTimeseries* document. We have collected these requirements from the most organizations most pressed by the needs for a standard, easily extendable solution.

7.6.1 Examples

One of the most important things when processing time series data is to be able to link the points of a light curve to the original image it was coming from. An outlier on the light curve could be a static noise or some other kind of error on the image but it could be also a valid burst in the object's magnitude. We can identify this on the first look if we have the link to the original data for each point of the light curve. The sample serialization of such provenance tracking can be seen on Fig. 13.

The implementation on server side is easy and we plan this to be a very simple operation on the client side too. Just one click on the point of the light curve can send the coordinates via SAMP to Aladin client which displays the original image. The light curve plotted within TopCat client can be seen on Fig. 14. With the metadata *Time Series Cube DM* provides, Topcat could automatically identify what to do when user clicks on a point of the light curve. In our case, we want to take the URL of the original image from which this point of the curve originates and send it to Aladin via SAMP, see Fig. 15. We are also able to send just a cutout instead of the whole image link, as seen on Fig. 16.

7.7 Gravitational waves

This example is quite similar to the one described in Chap. 7.5. The only difference is a temporal axis describing a frequency (as time derivative) and then another dependent axis that contains measured data for gravitational wave analysis. This use case is taken from the LIGO tutorial site (*Signal Processing with GW150914* (n.d.)).

In our specific uses case, we describe a time series data cube storing gravitational wave analysis, as can be seen on Fig. 18. We have a temporal axis of `frequency` rendered as `x axis` and the `strain/rtHz` dependent axis rendered as `y axis`. The second independent axis holding sampling rate frequency is rendered as 3rd dimension in colors.

Storing the sampling rate frequency as an independent axis would make sense if it would be a configuration of the instrument, theoretically providing different values at the same point of time when measuring on different frequencies. Otherwise its just another dependent axis (and a very primitive one, just a multiple of the values for different sampling frequencies).

```
1   <!--================== Cube annotation =======================-->
2
3 ▼ <INSTANCE ID="ndgnsoliggsa" dmtype="ndcube:Cube">
4       <ATTRIBUTE dmrole="independentAxis">
5           <COLUMN ref="HJD"/>
6       </ATTRIBUTE>
7 ▼     <ATTRIBUTE dmrole="dependentAxis">
8           <COLUMN ref="MAG"/>
9           <COLUMN ref="raj2000"/>
10          <COLUMN ref="dej2000"/>
11      </ATTRIBUTE>
12  </INSTANCE>
13
14  <!--================== VOTable columns =======================-->
15
16  <FIELD ID="HJD" datatype="double" name="HJD" ucd="VOX:Image_MJDateObs" unit="d">
17      <DESCRIPTION>Epoch at midpoint of observation in Heliocentric Modified Julian Date</DESCRIPTION>
18  </FIELD>
19  <FIELD ID="MAG" datatype="float" name="MAG" ucd="phot.mag" unit="mag">
20      <DESCRIPTION>Magnitude of the object</DESCRIPTION>
21  </FIELD>
22  <FIELD ID="raj2000" datatype="double" name="raj2000" ucd="pos.eq.ra;meta.main" unit="deg">
23      <DESCRIPTION>Observed RA of the object</DESCRIPTION>
24  </FIELD>
25  <FIELD ID="dej2000" datatype="double" name="dej2000" ucd="pos.eq.dec;meta.main" unit="deg">
26      <DESCRIPTION>Main value of declination</DESCRIPTION>
27  </FIELD>
28
29  <FIELD ID="SIAPLink"  arraysize="*" datatype="char" name="accref" ucd="meta.ref.url;datalink.preview">
30      <DESCRIPTION>Link to cutout of image</DESCRIPTION>
31  </FIELD>
32
33 ▼ <DATA>
34 ▼     <TABLEDATA>
35 ▼         <TR>
36              <TD>56287.11921047</TD>
37              <TD>19.0829</TD>
38              <TD>-72.9580158132637</TD>
39              <TD>19.0339</TD>
40              <TD>http://vos2.asu.cas.cz/getproduct/dk154_reduced/data/20121225/SMCf3_000001_sci.fits.fz?sdec=0.01&amp;dec=19.0339&amp;ra=-72.9580158132637&amp;sra=0.01</TD>
41          </TR>
42      </TABLEDATA>
43  </DATA>
44
45
```

Figure 13: Linkage towards original image cutout for one point of the light curve. The data can be seen in the last column of the table with cutout parameters taken from the right ascension and declination of this point of the curve. Note that while the cube could mention the `SIAPLiink` column as one of its axes, any statistical distribution or plotting of its values would not be meaningful. That is the reason why it is not part of the cube and is described entirely by the `FIELD` element that might be forged by the *Provenance DM* in the future.
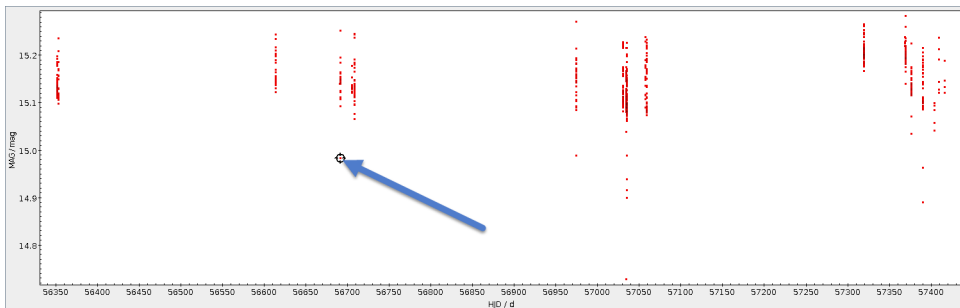

Figure 14: A light curve rendered in Topcat client. The graph represents magnitude based on the HJD when the star was observed. After clicking on the highlighted point, Fig. 15 or Fig. 16 follows.

We assume that we have the more complicated option here that different sampling frequencies can provide different measurements.

7.7.1 Examples

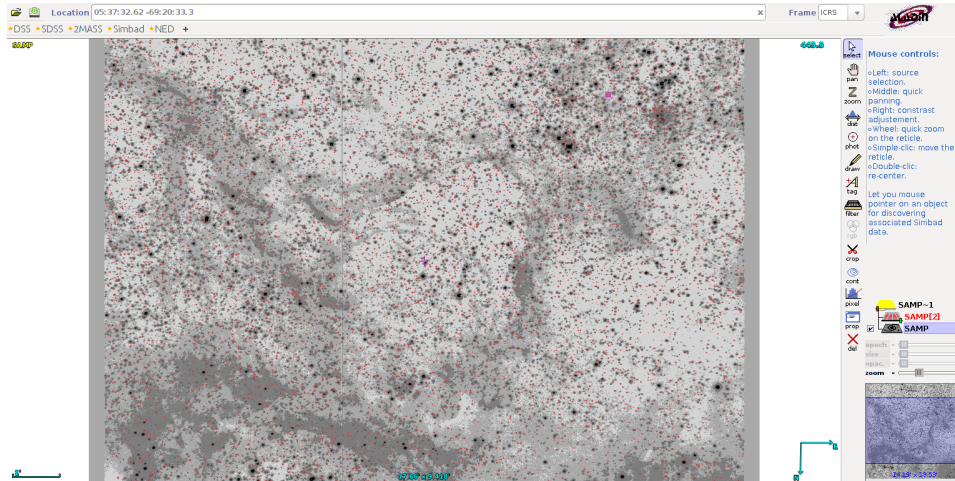The sample serialization and one row can be seen on Fig. 17.

24

Figure 15: Original image for one point of a light curve. After clicking on the point, the image from which it originates is displayed in Aladin.
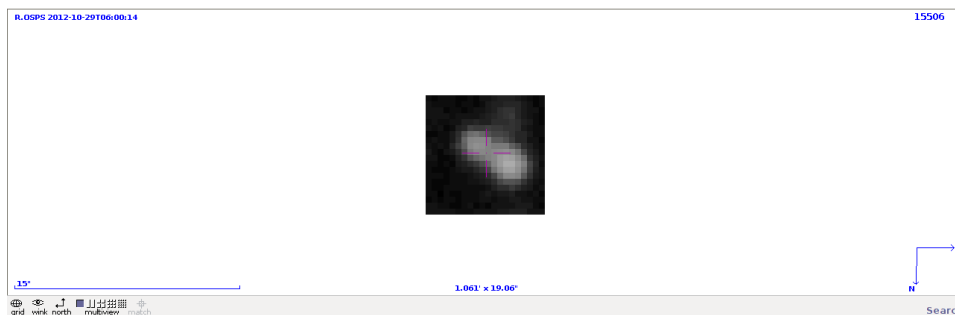

Figure 16: Small cutout from original image for one point of a light curve. If a SIA cutout link is sent instead of the SIA link to the whole image, Aladin will render it the same way as the whole original image.

```
 1  <GROUP utype="Cube.Char">
 2      <GROUP utype="NDPoint">
 3          <GROUP ucd="meta.main;em.freq" utype="TimeAxis">
 4              <FIELDref ref="Freq" utype="Coverage.Location.Coord"/>
 5          </GROUP>
 6          <GROUP ucd="meta.main;pos.distance" utype="ObservableAxis">
 7              <FIELDref ref="Strain" utype="strain"/>
 8          </GROUP>
 9      </GROUP>
10  </GROUP>
11
12  <!--================= Cube annotation =======================-->
13
14  <INSTANCE ID="ndgnsoliggsa" dmtype="ndcube:Cube">
15      <ATTRIBUTE dmrole="independentAxis">
16          <COLUMN ref="Freq"/>
17          <COLUMN ref="SamplingFreq"/>
18      </ATTRIBUTE>
19      <ATTRIBUTE dmrole="dependentAxis">
20          <COLUMN ref="hardnessRatio"/>
21      </ATTRIBUTE>
22  </INSTANCE>
23
24  <!--================= VOTable columns =======================-->
25
26  <FIELD ID="Freq" datatype="double" name="freq" ucd="meta.main;em.freq" >
27      <DESCRIPTION>Frequency</DESCRIPTION>
28  </FIELD>
29  <FIELD ID="SamplingFreq" datatype="double" name="sampling frequency" ucd="meta.main;em.freq" >
30      <DESCRIPTION>Sampling frequency</DESCRIPTION>
31  </FIELD>
32  <FIELD ID="Strain" datatype="float" name="strain" ucd="meta.main;pos.distance">
33      <DESCRIPTION>Strain of two mirrors</DESCRIPTION>
34  </FIELD>
35
36  <DATA>
37      <TABLEDATA>
38          <TR>
39              <TD>100</TD>
40              <TD>4096</TD>
41              <TD>1.05631e-23</TD>
42          </TR>
43      </TABLEDATA>
44  </DATA>
```

Figure 17: VOTable example for gravitational Wave data. The independent axes are here the frequency and sampling frequency (explained in Chap. 7.7. The sample row is showing a strain for 100 Hz frequency sampled with 4096 Hz.
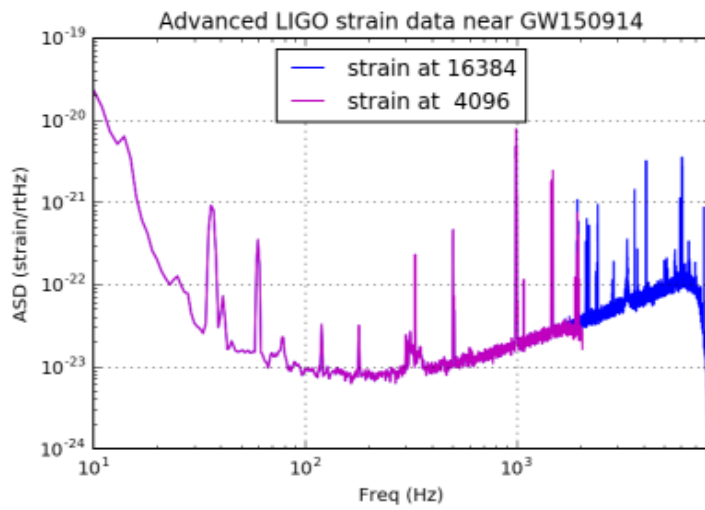
Figure 18: LIGO time series plotting example. This graph is showing strain for frequency between 10^1 Hz and 10^4 Hz for two different samplings of 4096 Hz and 16384 Hz which are expressed in colors.

8   Reference implementation

We are working on GAVO DaCHS implementation for server-side service providing time series data cubes in the *Time Series Cube DM* format. The service is ready and providing real data, but we have only a development environment ready right now and is under constant change. If you want to have a look on the final version, you can find it on the following web page (Nadvornik (2017)).

   We are also working on the SPLAT-VO modifications so it can understand the *Time Series Cube DM* format. Since we are using standard VOTables for serialization of our data cubes, the modifications are more related to SPLAT-VO understanding the axis domain models like magnitude axis inversion or time axis unit identification. In the current version of the serialization, SPLAT-VO understands the data perfectly.

9   Summary

This is still a work in progress and will need a lot of discussion and polishing to make it to an IVOA recommendation. However, we believe that this document creates a solid foundation upon which we can continue building.

References

Christophe Arviset, S. G. e. a. (2010), 'IVOA architecture', IVOA Note.
   URL: *http://www.ivoa.net/documents/Notes/IVOAArchitecture*

Doug Tody, M. D. e. a. (2012), 'Simple spectral access protocol version 1.1', IVOA Recommendation.
   URL: *http://www.ivoa.net/documents/SSA/20120210/REC-SSA-1.1-20120210.htm*

Douglas Tody, F. B. e. a. (2013), 'Ivoa image data model, version 1.0', IVOA Working Draft.
   URL: `http://wiki.ivoa.net/internal/IVOA/ImageDM/WD-ImageDM-20130812.pdf`

Efficient XML Interchange (EXI) Format, Version 1.0 (2014), W3C Recommendation.
   URL: `https://www.w3.org/TR/exi/`

et al., D. T. F. B. (2015), 'Ivoa n-dimensional cube model, version 1.0', IVOA Working Draft.
   URL: `http://www.ivoa.net/documents/NDimCubeDM/20150320/WD-CubeDM-1.0-20150320.pdf`

Gerard Lemson, O. L. e. a. (2016), 'Vo-dml - a consistent modeling language for ivoa data models', IVOA Proposed Recommendation.
URL: *http: // www. ivoa. net/ documents/ VODML/ 20160923/ VO-DML-PR-v1. 0. pdf*

Jesus Salgado, C. R. e. a. (2013), 'Ivoa photometry data model', IVOA Recommendation.
URL: *http: // www. ivoa. net/ documents/ PHOTDM/ 20131005/ REC-PhotDM-1. 0-20131005. pdf*

Kimball, R. (1997*a*), 'A dimensional modeling manifesto', Article.
URL: *http: // www. kimballgroup. com/ data-warehouse-business-intelligence-resources/ kimball-techniques/ dimensional-modeling-techniques/ star-schema-olap-cube/*

Kimball, R. (1997*b*), 'A dimensional modeling manifesto', Article.
URL: *http: // www. kimballgroup. com/ 1997/ 08/ a-dimensional-modeling-manifesto/*

Matthew Graham, E. S. e. a. (2014), 'Employing simpletimeseries for representing time series', IVOA Note.
URL: *http: // www. ivoa. net/ documents/ Notes/ SimpleTimeSeries/ 20140513/ NOTE-SimpleTimeSeries-1. 0-20140513. html*

Mireille Louys, D. T. e. a. (2011), 'Observation data model core components, its implementation in the Table Access Protocol, version 1.1', IVOA Proposed Recommendation.
URL: *http: // www. ivoa. net/ documents/ ObsCore/ 20111028/ REC-ObsCore-v1. 0-20111028. pdf*

Nadvornik, J. (2017), 'Aiascr vo services', Web page.
URL: *http: // vos2. asu. cas. cz/*

Patrick Dowler, D. T. e. a. (2015*a*), 'Ivoa simple image access, version 2.0', IVOA Recommendation.
URL: *http: // www. ivoa. net/ documents/ SIA/ 20151223/ REC-SIA-2. 0-20151223. pdf*

Patrick Dowler, F. B. e. a. (2015*b*), 'IVOA datalink', IVOA Recommendation 17 June 2015.
URL: *http: // www. ivoa. net/ documents/ DataLink/*

Petr Skoda, P. W. D. e. a. (2014), 'Spectroscopic Analysis in the Virtual Observatory Environment with SPLAT-VO', *ArXiv e-prints* , arXiv:1407.1765.
URL: *https: // arxiv. org/ abs/ 1407. 1765v1*

Rots, A. (2007), 'Space-time coordinate metadata for the virtual observatory, version 1.3', IVOA Recommendation.
  URL: *http: // www. ivoa. net/ documents/ latest/ STC. html*

Signal Processing with GW150914 (n.d.), IPython Notebook.
  URL: *https: // losc. ligo. org/ s/ events/ GW150914/ GW150914_ tutorial. html*

Solano, E. (2012), 'Science use cases for time series', Science use Cases for Time Series.
  URL: *http: // wiki. ivoa. net/ twiki/ bin/ view/ IVOA/ CSPTimeSeries*