International

Virtual

Observatory

Alliance

Characterisation of an astronomical image

Version 1.00

IVOA Note 2007 May 08

This version:
1.00-20070508
Latest version:
http://www.ivoa.net/Documents/latest/latest-version-name
Previous version(s):

Author(s):
Alberto Micol, Francois Bonnarel, Igor Chilingarian, Mireille Louys, Jonathan Mc Dowell, Anita Richards

---

Abstract

The goal of this note is to illustrate how typical metadata pertaining to an astronomical image, being it product of an observation or of a simulation, can be mapped to elements of the Characterisation Data Model version 1.1 (hereafter CharDM [1]).

By explaining the IVOA Characterisation Data Model concepts via a common and concrete example, we hope to help the data providers, the IVOA standard developers, and even ourselves, to understand the model, its completeness, and its usefulness.

Status of This Document

This is a Note. The first release of this document was 2007 May 08 (this one).

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/.

Acknowledgements

We would like to thank all the many people that participated into various discussions about data models in general, and more specifically the ones who pushed us into deeper waters from which we emerged refreshed, like the ST-ECF, the STScI and MAST, the CADC, the ESO-VOS, and the ESAC-VO groups.

Contents

Page intentionally left blank

1 Introduction

Exemplifying, this document tries to answer to questions like:

Which CharDM element maps to the exposure time?

This question will be extended to each typical image metadata –think of it as a (set of) header keyword(s). The answer will be given using the concept of **UTYPE** ("unique type"), which in VO-speak is a unique identifier to any data model element.
To carry on the exposure time example, the answer to the previous question is:

The exposure time maps to the characterization data model element pointed to by the UTYPE: ***timeAxis.coverage.support.extent***

where one can see that the utype consists of number of tokens concatenated using a dot; in the majority of the cases, and not always, the utype will be composed of four tokens, in sequence: the "*axis"* onto which a "*property"* is described at a given "*level"* of detail, using agreed "*attributes"* to describe a given concept. In other cases, less than 4 tokens are given, usually to provide some flag, or some "status", as in: on a given "axis" a certain "status" is provided (e.g. the calibration status could be "calibrated" or "uncalibrated", etc.). Typically, the utypes in this document will be in one of the following forms:

UTYPE's form:	*Example:*
axis.status	spatialAxis.calibrationStatus
axis.property.status	spatialAxis.samplingPrecision.undersampling
axis.property.level.attribute	spectralAxis.coverage.bounds.limits

It is worth noting that not all the typical metadata will find a corresponding element in the CharDM, simply because CharDM cannot, nor should, describe every possible metadata associated to an observation. CharDM only describes the N-dimensional space the observation is immersed into, but this will be said later, in Section 2.

We will go through the four different axes (spatial, spectral, temporal, and flux), and on each we will describe the most common "header keywords" (so to say) that data providers are used to, and map them to CharDM UTYPEs.

Given that the CharDM does not cover (nor it should) all possible aspects of an observation (e.g. it does not describe the gain used by the detector amplifier), we will explicitly mark those metadata that do not belong to CharDM, and/or we will prefix them with a particular string (the "namespace", e.g., the string "STC:" for the Space Time Coordinates [2]) to refer the data providers to any other Data

Model that could be relevant. Non-prefixed UTYPEs default to the CharDM data model.

Structure of the document

Section 2 will describe the CharDM model at a very high level.
Section 3,4,5 and 6 will describe the 4 axes of the 2D image, respectively the spatial, spectral, temporal, and the flux axes. On each axis the typical 2D image metadata will be mentioned, and, if possible, the mapping to the proper UTYPE will be given, along with its compliance status (Mandatory, Required, Optional) in square brackets.

2 The Characterisation Data Model in few words

N-dimensional space subtended by an observation

The model is meant to describe, possibly in physical units (as opposed to detector/instrument internal units), the N-dimensional space subtended by an observation (being it synthetic or real), and/or any of its products, with the exclusion of catalogues whose characterization is going to be described by a different model (the *source catalogue* data model). The model is not to describe how the observation has been achieved or which targets have been observed.

Axes of the N-dimensional space

Possible axes of this N-dimensional space are: spatial, spectral, temporal, and the observable –typically flux- which is the quantity being measured by the instrument. Polarimetry and radial velocity are other possible axes.

For example, a 2D image, like a DSS image, is immersed in a four dimensional space constituted by the spatial (bi-dimensional) axis, by a spectral axis, the temporal axis, and the flux axis; a total of 4 axes in this case (one of which, the spatial, is bi-dimensional). It is evident that the classical name “2D image” has more to do with the way the product is packaged, than with the actual space it covers. A 1D spectrum (e.g. flux vs. wavelength) is also immersed in the same four-dimensional space.

Properties described along the given axes

Once immersed in the above-described N-dimensional space, the typical questions that come to mind when measuring a signal (typically photons in our case) are:

Where does the signal (e.g. photons) come from?

In this model we call such property Coverage. **Coverage** typically describes which part of the sky, which part of the electromagnetic spectrum, and at which times the photons were collected. But it can also describe on the observable axis how deep an image is.

How is the observation sampled?

The **SamplingPrecision** property describes the sampling on all the axes; for example, pixel scales, bin sizes, etc.

Can two events closer than a given offset be resolved?

The **Resolution** property describes that.

Detailing descriptions as much as possible

Depending on the science case, to know that an HST image was taken with a red filter might suffice. In other cases the actual transmission curve of the combined optics, filter, and detector is required.
Each of the above-mentioned properties can be described at various levels of detail. We call these levels: Location, Bounds, Support, Sensitivity or Variability.

Location is the coarsest level, where only a reference position of a point of the N-dimensional space will be given. This point does not carry any particular meaning -it is not precisely defined-, it is just meant to roughly tell where the observation is located on each of the axes.

Bounds describe not just a point but a N-dimensional box into which the data are contained. In the Coverage sense, the promise is that all the relevant photons were actually coming from a region all inside this N-D box. In the Resolution sense, the bounds are to describe the possible range of resolution values.
Worth noting that it is useful to specify the bounds in all cases, even if the represented quantity (e.g. resolution) is constant! Without explicit bounds it might be impossible for the reader to recon that the quantity is actually constant.

Support is where more detailed geometrical information can be described. E.g.:

- the slit position onto the sky (spatialAxis.coverage.support for 1D spectra),
- the footprint of an image (spatialAxis.coverage.support for 2D images)
- an array of time intervals describing the epochs the photons were collected from (timeAxis.coverage.support)

Sensitivity or Variability is reserved to a next version of the model, whereby variations along the axes of the various quantities will be provided. Examples:

- Exposure map
- Transmission curve, to detail the spectral sensitivity
- Resolution maps, to detail the variation of the resolution along some axis
- Sampling maps, to detail the variations of the pixel scale along the axes, for non-equally sampled data

In the present model the four levels of detail are identified by different names across the different properties.

Last but not least, please keep in mind that the CharDM model serves two purposes: data discovery and serialisation. Both are fundamental to the VO.
Data Discovery: A common and agreed way to identify pieces of information (like the astrometric error of an image, its start time, its waveband, etc) is central to the data discovery process when not just one service is searched for valuable scientific data, but the entire VO is scanned through.
Serialisation: An agreed format is essential to deliver the search results in a interoperable way.

3 The Spatial Axis

Concepts to be covered on this axis are:

- Units
- Coordinate system;
- Spatial coordinates, including the region of the sky from which photons were collected
    - (e.g. footprint information)
- Astrometric accuracy (e.g., in arcsec)
- Astrometric quality (was it calibrated or not?)
- Spatial resolution

3.1 Spatial units

The CharDM element corresponding to the units used for all the quantities on the spatial axis is:

spatialAxis.unit **[M]**

In the model, all quantities on the spatial axis inherit the units referred to by that UTYPE. This does not imply that all properties and/or levels should use the same units. It is always possible to specify different units within each property/level.

Correspondingly, the UTYPE to be used e.g. for resolution can be explicit with the element whose utype is**: spatialAxis.resolution.unit**.

3.2 Coordinate system

Each axis defines its own coordinate system. For the spatial axis this is referred to by the UTYPE:

spatialAxis.coordsystem **[M]**

Typical values are FK5, ICRS, etc. As for the units, the same inheritance mechanism applies.

3.3 Where on the sky?

There are various parameters that could be used to describe the positioning on the celestial sphere of an image:

- Sky coordinates (CRVAL1 and CRVAL2) of a given reference pixel
- Roll angle (the position angle North to East of the Y axis)
- Coordinates of the main object being observed
- Size of the field of view (e.g., 160 arcsec)
- Sky coordinates of the 4 corners of a “rectangle” inscribing the image
- N polygons, one for each of the N chips, detailing the exact field of view of each chip

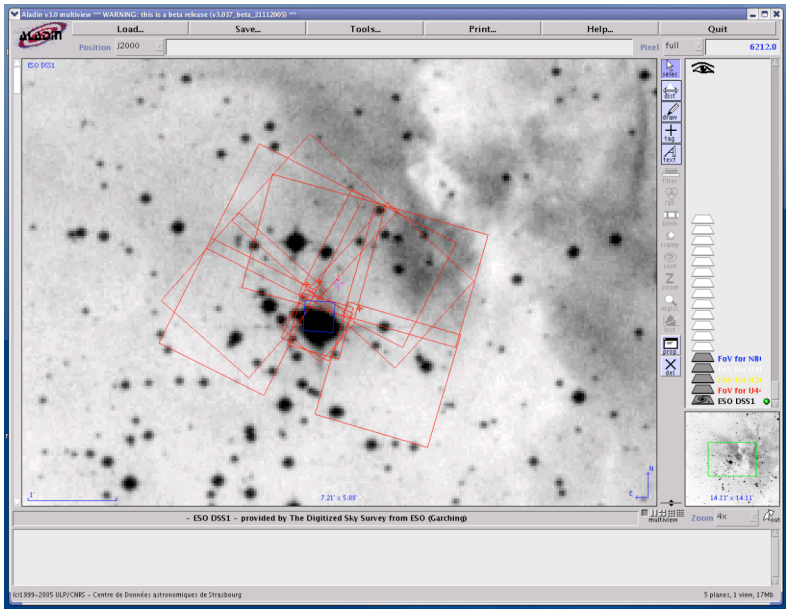

Figure 1: Footprints around Eta Carinae

3 HST/WFPC2 (4 chips, red) and 1 HST/NICMOS (1 chip, blue)

The characterization data model does not describe instrumental setup or observer selected target. Information like “coordinates of main object” (eta carinae in Figure 1) is not described by CharDM.
The footprints, that is the exact regions from which the photons were collected, is what CharDM likes to describe. That is, the N polygons case mentioned above (see Figure 1). The place where this is described in the CharDM is:

spatialAxis.coverage.support.area[] **[R]**

The ending square brackets are used to indicate that a number of parameters are to be provided in this case. (Specifically, the serialization of such element should obey to the STC:AstroCoordArea Region(2D) polygon specification).

If the exact region is unknown, indication of the spatial coverage can be given using the rougher

spatialAxis.coverage.bounds.limits[] **[R|M]**

which only provides a broad sky box, with the only promise that the photons came from somewhere inside that box. Typically this is described using the minimum and maximum of the right ascensions and of the declinations (or of the spatialAxis.coverage.bounds.coordsystem, [or spatialAxis.coverage.coordsystem, [or spatialAxis.coordsystem]]) used. (This is serialized using the STC:CoordArea specification whereby ranges on individual spatial axes are OR-ed together.)

If even the bounds are not known, then a rough indication of where the photons were coming from should be given. That can be done using:

spatialAxis.coverage.location.coord[] **[R|M]**

For theoretical or simulated images, it is not required to provide any spatial coverage. This can be made explicit setting the value of the element spatialAxis.coverage.coordsystem should be set to RELOCATABLE. Please refer to the coordsys RELOCATABLE token in the Spectrum data model for in-depth description.

It is possible to provide the roll angle (do not forget that it must be given with the same units as defined for spatialAxis.coverage.bounds) via:

spatialAxis.coverage.bounds.limits.charBox.size2.PosAngle **[O]**

It is possible to provide an indicative linear size of the field of view using:

spatialAxis.coverage.bounds.extent **[O]**

or

spatial.coverage.bounds.limits.charBox.size2 **[O]**

for describing the field of view sizes along the two directions.

3.4 Astrometric calibration

As on any other axis, it is possible (if relevant) specify whether the information on that axis has been calibrated or not. Please refer to the CharDM document for explanations.

The element in the model that provides this information is:

spatialAxis.calibrationStatus **[R]**

and could assume only one of the following values: CALIBRATED, UNCALIBRATED, RELATIVE, NORMALIZED. For example, if your image was astrometrically calibrated using a reference catalog (USNO, GSC, etc), that can be indicated declaring calibrationStatus := CALIBRATED and not RELATIVE[1]!

3.5 Astrometric accuracy

The error on the spatial coverage is covered by the container elements:

spatialAxis.accuracy.statError.errorRefVal **[M]**
spatialAxis.accuracy.sysError.errorRefVal **[M]**

These are only containers by which it is possible to specify both the statistical and the systematic error components: their values using the sub-element errorRefVal.errorRefValue, and their optional documentation, using errorRefVal.documentation. For example, if GSC2.3 catalog was used to astrometrically calibrate the image, it is possible to specify the statistical deviations from the GSC solution using the statError elements; also, if you do not expect any other systematic effect, you could use the sysError for the known overall precision of the GSC catalog.

3.6 Spatial resolution

The spatial resolution can also be provided at various levels of details. Typical resolution values are just given using the resolutionRefVal level:

spatialAxis.resolution.resolutionRefVal.resolutionRefValue **[R|M]**
spatialAxis.resolution.resolutionRefVal.documentation **[O]**

[1] RELATIVE would mean that your image is not described using standard coordinates, like RA and DEC, but instead uses offsets (in arcsec) from a given position; very unlikely!

We encourage to provide also the resolution bounds [Recommended by CharDM], using the element:

spatialAxis.resolution.resolutionBounds **[R|M]**

That is useful to indicate whether the resolution is expected to vary within the image, or otherwise to explicit that the resolution is actually constant in the field of view, by setting the bound limits to the same value.

3.7 Pixels: spatial sampling

The typical spatial sampling information accompanying a 2D image describe the number of pixels, and the pixel scale(s).

The number of pixels for a 2D image is associated to the element:

spatialAxis.numBins **[R]**

which can be serialized using any of the NumBins, NumBins2, NumBins3 subtypes.

The pixel scale is to be provided using:

spatialAxis.samplingPrecision.samplingRefVal.samplingPeriod [M]

The sampling period is defined as the distance from center to center of two adjacent bins (it might include a gap between the two bins, hence it would be bigger than the bin size).
The bin size element instead is:

spatialAxis.samplingPrecision.samplingRefVal.samplingExtent [R]

which could provide the pixel size in microns.

The CharDM offers the possibility to explicitly state whether the sampling is regular or not, or whether the data are affected or not by undersampling. Please use the

SpatialAxis.regularSamplingStatus (True or False) **[O]**
SpatialAxis.undersamplingStatus (True or False) **[O]**

In case the image has really important sampling variations (geometrical distorsions), the regularSamplingStatus will obviously set to False, and in such case it would be useful to provide actual ranges of variations of the pixel scale. Please consider using the elements under:

spatialAxis.samplingPrecision.samplingPrecisionBounds[2] [O]

to specify the minimum and maximum of the pixel scale values (with min = max if the pixel scale is constant).

4 The Spectral Axis

Concepts to be covered on this axis are:

- Units
- Spectral coordinate system (TOPOCENTER, HELIOCENTER, etc.)
- Spectral coordinates
- Wavelength calibration
- Wavelength accuracy
- Wavelength resolution
- Wavelength sampling (num of bins, bin size)

4.1 Spectral Units

Very symmetrically to the spatial axis, the CharDM element corresponding to the units used for all the quantities on the spectral axis is:

spectralAxis.unit **[M]**

In the model, all quantities on the spectral axis inherit the units referred to by that UTYPE. This does not imply that all properties and/or levels should use the same units. It is always possible to specify different units within each property/level. Correspondingly, the UTYPE to be used will point to the unit element of that property/level; e.g**., spectralAxis.resolution.unit**.

4.2 Coodinate system: where on the spectrum?

The spectral coordinate system is expressed in the model as:

spectralAxis.coordsystem **[M]**

The default value for the spectral coordinate system is TOPOCENTER, meaning that no correction to the wavelength axis was applied to the data. (HELIOCENTER and other possible values for a 1D-spectrum are listed in the SpectrumDM document.)

[2] Valid sub-elements of the spatial sampling bounds are: unit, coordsystem, samplingPeriodLimits (here the min and max of the pixel scales), samplingExtentLimits, and documentation.

4.3 Wavelengths (Frequencies, Energies): spectral coordinates

The spectral coordinate of an image -that is, the spectral region covered by the overall transmission curve of the entire optical system (e.g. the convolution of the filter ⊗ detector ⊗ optics transmission curves, including atmospheric terms, etc.)- can be expressed in wavelength, frequency, or energy. To declare which type of coordinate is provided, the UCD, or "Uniform Content Descriptor", must be used.

This particular UCD element (each axis has a UCD element) in the CharDM is identified by the UTYPE:

spectralAxis.ucd **[M]**

Typical values will be: em.wl, em.freq, and em.energy for the three above-mentioned flavours of spectral coordinates.

A typical reference value for the entire image could be given using the CharDM element designated by the UTYPE:

spectralAxis.coverage.location.coord **[R|M]**

This value is not precisely, or mathematically, defined; it represents just a typical value around which photons are collected.

Wavelength (or frequency, or energy) Min and Max:
A little more precise information could be given by specifying the minimum and maximum values of the spectral coordinate. This can be done using the bounds mechanism already seen for the spatial axis:

spectralAxis.coverage.bounds.limits[] **[R|M]**

whereby it is declared that the entire transmission curve lays between those two values of the spectral axis. No guarantee is given that the entire interval is covered by flux information (e.g., there could be gaps).

To specify detailed information regarding the spectral regions, within the bounds, which actually add to the image, it is possible to provide an array of intervals into which the transmission curve is defined and non-zero (even the case of a three color image is supported), via the element:

spectralAxis.coverage.support.area **[M]**

(It can be serialized using the ranges as specified by STC:CoordArea).

4.4 Wavelength calibration: spectral axis calibration

In general, though probably of little use in a 2D image case, the status of calibration on the spectral axis can be provided using:

spectralAxis.calibrationStatus **[R]**

Valid values are the same as for the spatial axis: CALIBRATED, UNCALIBRATED, RELATIVE, NORMALIZED. See 3.4 above.

4.5 Errors on wavelength calibration: spectral axis accuracy

The accuracy with which the wavelength, frequency, or energy is known can be expressed using the CharDM elements:

spectralAxis.accuracy.statError **[R]**
spectralAxis.accuracy.sysError **[R]**

Again, this is probably not very useful in the imaging case, while it is essential if we were to describe a spectrum.

4.6 Spectral resolution

Not Applicable to the 2D Image case.

4.7 Bin size: spectral sampling

Not applicable to the 2D image case.

5 The Time Axis

Let's suppose that the simple image to be described is actually the product of a co-addition of two or more individual images. The concepts one may want to describe are: the total exposure time, an indicative time instant for the imaging product, the first of the start times and the last of the stop times, but it can be useful to also provide more information like the start time and stop time of each individual image, for precise temporal coverage. Let's first start introducing the axis, its units and its coordinate system.

5.1 Time units

Again, as for the spatial and spectral axes, the CharDM element corresponding to the units used for all the quantities on the time axis is:

timeAxis.unit **[M]**

In the model, all quantities on the time axis inherit the units referred to by that UTYPE. Though, very likely the start time will make use of modified julian days,

or of the ISO-8601 convention, while the total exposure time will be expressed in seconds. It is possible to declare a different unit at any property or level. E.g**., timeAxis.coverage.support.unit**.

5.2 Time coordinate system

The temporal coordinate system is expressed in the model as:

timeAxis.coordsystem **[M]**

The coordinate system declaration must adhere to the Space Time Coordinates data model (STC [2]). A TimeFrame is composed of a reference position in space[3], and a time scale[4]. You can use TOPOCENTER to declare that measured times refer to the space location of the observatory itself (spatialAxis.coverage.obsyLoc can be used for to define that). See the STC [2] data models to assign proper time scale values (UTC, TT, TAI, TCB, etc).

5.3 Time coordinates: start and stop times

In the most common case of a single, not time resolved, 2D image it might suffice to provide just an indication of the time at which the observation took place: such single value information can be provided as:

timeAxis.coverage.location.coord **[R|M]**

where the instant of time might be the mid point between the start and the stop times. Though, we encourage to provide a bit more detailed information. Again, if the product to be described is a single 2D image, it is useful to provide the start and the stop times, using the bounds level of the time coverage property:

timeAxis.coverage.bounds.limits[] **[R|M]**

Obviously, for a 2D image result of the co-addition of multiple images, it is better to provide the start and stop times of each and every observation that was used to generate the co-added product. The model allows that using the support level of the time coverage property:

timeAxis.coverage.support.area[] **[R]**

In such case (co-addition), the above-mentioned element (timeAxis.coverage.bounds.limits) is used to describe the overall time interval within which all observations were acquired. Basically this means that the “limits”

[3] The reference position could be the location of the observatory, or it could have been corrected to the barycentre of the solar system. In some applications it is useful to take into account the relativistic time’s contraction when correcting.

[4] Plus, but very rarely, a Time Reference Direction is also relevant. Please refer to [STC].

element is filled with the minimum of all the start times, and the maximum of all the stop times.

5.4 Time window: time coverage bounds extent

It is useful, especially from a query point of view, to describe the overall time span of a product composed of various observations. As an example, the overall time span of an image product, which observed a variable source at various times, is useful to indicate the baseline onto which source variability can be studied. The time span can be given e.g., in seconds as the time difference between the maximum of the stop times and the minimum of the start times. We call this

timeAxis.coverage.bounds.extent **[R]**

It is obviously provided at the bounds level, given that it can be computed from the timeAxis.coverage.bounds.limits information.

5.5 Exposure time: time coverage support extent

While for a 2D single image usually the exposure time coincides with the time span, the exposure time cannot be conveyed by the bounds extent, because the exposure time is a different concept than the time span. The accurate exposure time requires a more detailed level of information than the bounds can offer: the support level. This is immediately clear in the case of a co-added image, whereby the bounds extent is clearly different than the total exposure time. The total exposure time is provided by the element:

timeAxis.coverage.support.extent **[M]**

and can typically be computed by summing up the time intervals described by the time coverage support intervals, maybe weighted by a filling-factor[5].
It remains clear that more precise information can only be provided using an exposure map, which can be described going another level down in th CharDM structure, the **Sensitivity** level, not yet covered by this tutorial.

6 The Flux Axis

The model does not only describe the independent axes of the observations (typically spatial, spectral, and temporal axes) but also the axis (or axes) associated to the measured quantity, most typically a Flux. Such quantity can be

[5] The filling factor can be provided using the utype:
timeAxis.samplingPrecision.samplingPrecisionRefVal.fillFactor

described using the same information as on any other axis. The fact that this is a dependent axis can be stated using the element:

fluxAxis.independentAxis = FALSE **[R]**

while the corresponding element on the other axes, that is, spatialAxis.independentAxis, spectralAxis.independentAxis, and timeAxis.independentAxis, should all be given value TRUE.

6.1 Units

The units to be used on this axis shall be provided, very similarly to the other axes, using the element:

fluxAxis.units **[M]**

6.2 Minimum and Maximum Flux

It is useful to provide the minimum and maximum values for the flux; that can be done using:

fluxAxis.coverage.bounds.limits[] **[R|M]**

6.3 Flux calibration status

Whether the flux values have been calibrated or not, and whether the measured values have been normalized, can be stated using the element:

fluxAxis.calibrationStatus **[R]**

and could assume only one of the following values: CALIBRATED, UNCALIBRATED, RELATIVE, NORMALIZED. The value CALIBRATED can be used if proper zeropoints (possibly with associated errors) are defined in the image header.

6.4 Calibration Accuracy

While the flux errors “per pixel” might be available in the data (weight maps, again to be demanded to the Sensitivity layer not described here), the characterization data model offers the possibility to describe the typical flux error of the entire product, distinguishing between the statistical flux error and the systematic error. The container element

fluxAxis.accuracy.statError.errorRefVal **[R]**

can host one or two sub-elements, the error value and its documentation:

fluxAxis.accuracy.statError.errorRefVal.errorRefVal [R]
fluxAxis.accuracy.statError.errorRefVal.documentation [O]

Similarly for the systematic error:

fluxAxis.accuracy.sysError.errorRefVal.errorRefVal [R]
fluxAxis.accuracy.sysError.errorRefVal.documentation [O]

6.5 Signal to Noise Ratio

The signal to noise ratio is not (yet) covered by the CharDM.

Appendix A: Summary of above-described metadata

SPATIAL AXIS

Independent axis:
spatialAxis.independentAxis = TRUE (for 1D spectrum)

Spatial coordinate system:
spatialAxis.coordsystem

Spatial units (typically deg):
spatialAxis.unit

UCD for the spatial axis:
spatialAxis.ucd

Footprint coordinates:
spatialAxis.coverage.support.area[]

Astrometric accuracy:
spatialAxis.accuracy.statError.errorRefVal
spatialAxis.accuracy.sysError.errorRefVal
spatialAxis.calibrationStatus
spatialAxis.resolution.resolutionRefVal.referenceValue
spatialAxis.resolution.resolutionRefVal.documentation

SPECTRAL AXIS

Independent axis:
spectralAxis.independentAxis = TRUE (for 2D image)

Spectral coordinate system:
spectralAxis.coordsystem

Spectral units:
spectralAxis.unit

UCD for the spectral axis (e.g., em.wl, or em.freq):
spectralAxis.ucd

Spectral wavelength intervals (freq, energy), where Flux is provided:
spectralAxis.coverage.support.area

Has the spectral axis been calibrated?
spectralAxis.calibrationStatus

Error with which the wavelength (freq., en.) information is known:
spectralAxis.accuracy.statError
spectralAxis.accuracy.sysError

Spectral resolution reference value (and documentation):
spectralAxis.resolution.resolutionRefVal.ReferenceValue
spectralAxis.resolution.resolutionRefVal.documentation

Spectral resolution min and max values:
spectralAxis.resolution.resolutionBounds.limits[]

Variation of the spectral resolution along the spectral axis:
spectralAxis.resolution.resolutionVariability.resolutionMap

Number of spectral bins:
spectralAxis.numBins

Spectral bin size:
spectralAxis.samplingPrecision.samplingRefVal.samplingExtent

Spectral scale from centre to centre:
spectralAxis.samplingPrecision.samplingRefVal.samplingPeriod

Spectral sampling regular or irregular?
spectralAxis.regularSamplingStatus (TRUE|FALSE)

Spectral sampling: undersampled or not?
spectralAxis.undersamplingStatus (TRUE|FALSE)

TIME AXIS

Independent axis:
timeAxis.independentAxis = TRUE (for 1D spectrum)

Time coordinate system:
timeAxis.coordsystem (See STC document [2])

Time units:
timeAxis.unit

Time intervals when shutter was opened:
timeAxis.coverage.support.area[]

Time span (max stop times - min start times):
timeAxis.coverage.bounds.extent

Total Exposure Time:
timeAxis.samplingPrecision.support.extent

FLUX AXIS

Independent axis:
fluxAxis.independentAxis = FALSE

Flux units:
fluxAxis.units

Flux calibration (was it calibrated? is it normalised?):
fluxAxis.calibrationStatus (CALIBRATED, UNCALIBRATED,
NORMALIZED, RELATIVE)

Flux accuracy:
fluxAxis.accuracy.statError.errorRefVal
fluxAxis.accuracy.statError.errorRefVal.errorRefVal
fluxAxis.accuracy.statError.errorRefVal.documentation
fluxAxis.accuracy.sysError.errorRefVal.errorRefVal
fluxAxis.accuracy.sysError.errorRefVal.documentation

Signal to Noise Ratio:
Not (yet) defined.

Appendix B: List of CharDM UTYPEs

To access the full list of CharDM UTYPEs supported by this version of the model, along with their description and status, please refer to:

http://www.ivoa.net/Documents/UtypeListCharacterisationDM1.1.html

References

[1] J. McDowell, F. Bonnarel, I. Chilingarian, M. Louys, A. Micol, A. Richards, *Characterisation Data Model v1.1*, http://www.ivoa.net/Documents/latest/chardm.html

[2] A. Rots, *Space Time Coordinate data model (v1.30),* http://www.ivoa.net/Documents/latest/STC.html

[3] J. McDowell, F. Bonnarel, I. Chilingarian, M. Louys, A. Micol, A. Richards, *Utype list for the Characterisation Data Model (v1.1*), http://www.ivoa.net/Documents/UtypeListCharacterisationDM1.1.html

[4] R. Hanisch, M. Dolensky, M. Leoni, *Document Standards Management: Guidelines and Procedure* , http://www.ivoa.net/Documents/latest/DocStdProc.html