International
Virtual
Observatory
Alliance

Handling Units in the VO Version 0.2

IVOA Working Draft May 22, 2009

This version:
http://www.ivoa.net/internal/IVOA/UnitsDesc/WD-VOUnits-v0.2-20090519.pdf

Latest version:
http://www.ivoa.net/internal/IVOA/UnitsDesc/WD-VOUnits-v0.1-20090519.pdf

Previous versions:
0.1

Editor(s):
Anita Richards, Sébastien Derrière, Mireille Louys.

Authors:
François Bonnarel, Sébastien Derrière, Pedro Osuna,
Bruno Rino, Jesus Salgado, François Ochsenbein

Contents

Abstract

This document describes common practices in manipulating units in astronomical metadata and will define a means of consistent representation within VO services.

1 Status of this document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or rendered obsolete by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

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

Acknowledgements

We thank all those participants in IVOA and EuroVO workshops who have contributed by exposing use cases and proving comments, especially Paddy Leahy, Jeff Lusted, Arnold Rots, Mark Taylor, Brian Thomas and recent contributors on the DM forum.

2 Introduction

This document describes a model for "The use of Units in the VO" (hereafter simply "VOUnits"). In this Introduction we present an overview of this model in the context of the requirements of the VO and its users.

This document is not finished and we seek comments and volunteers, especially in examining the suitability of the proposed Standards 4 and how they will be implemented.

In Section 3 we provide links to existing specifications and conventions for the use of Units labels. Section 4 suggests the most convenient conventions for the IVOA to adopt internally, Section 5 outlines some use cases and Section 6 presents examples of how this could be implemented.

Generally, every quantity provided in astronomy has a unit attached to its value or is unitless (e.g. a ratio, a numerical multiplier).

The VO and its users employ units:

- in IVOA protocols, while exchanging XML documents, in various formats (VOTable, XML)
- in applications (input and output)
- in queries (input and output)

Units can be embedded in data (e.g. FITS headers) or be implied by convention and/or (preferably) specified in metadata.

2.1 The purpose of the Units model

The need for a Units model and a Units controlled vocabulary has arisen from several needs exemplified below:

1. To understand `m`, `meter`, `metre` ... as the same thing
2. To differentiate between `mm` (milli-metre) and `m.m` (`m2`)
3. To distinguish between the use of m as a wavelength (= c/frequency) and `m` as a distance e.g. `1 AU = 1.499 1011 m`
4. To create combinations of units intelligently e.g. `J.s-1 = W`
5. To translate between SI prefixes (`G, M, k` etc.) intelligently

This model is **not** prescribing what units data providers employ. So long as data are labelled in a recognised system, a translation layer will be provided. Data providers can customise the translation tools if required. Depending on preference and the operations required, the user may have a choice of units for their query and for the result.

It is **not** the role of this work package to provide a means of converting instrument-related units such as magnitudes, nor to perform coordinate conversions. The former is the domain of the PhotometryDataModel to be issued in a separate Data Model effort. See (`http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/PhotometryDataModel`) for more information and current developments.
Excellent libraries e.g. **AST** in C [AST] or **SLALIB** for Java, already exist for the latter.

It **is** our role to meet the needs of these work packages and provide consistency, as well as ensuring that the tools supporting simple conversions (not requiring external or conditional information) are available.

3 Current use of units

3.1 Vocabularies

In the long term this list may more usefully be maintained on a separate web page.

Many other projects have already produced lists of preferred representations of units. We give links to those most commonly used in astronomy (*please notify us of omissions*).

Unit labels suitable for IVOA use (or other electronic manipulation) need to be expressed unambiguously in encodings recognised by XML parsers, Java compilers, etc. At this stage, we will not attempt to deal with representations which are only suitable for use in human-readable literature, e.g. many IAU symbols using Greek letters. We use italics for physical constants for convenience in the parts of this document aimed solely at humans, but this cannot be used to distinguish labels in use (speed of light c is the same as *c*).

- `http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/IAUVizieRHEASARCGNU`
  Comparison between IAU recommendations, VizieR, HEASARC and the GNU software Units (S.Derrière)
- `http://vizier.u-strasbg.fr/doc/catstd-3.2.htx`
  Standards for Astronomical Catalogues, Version 2.0
- `http://arxiv.org/pdf/astro-ph/0511616`
  Dimensional Analysis applied to spectrum handling in VO context (Osuna & Salgado 2008) offers a mathematical framework to guess and recompute SI units for any quantity in astronomy. *Are the library and tools used available outside VOSpec, for incorporation in other tools?*
- `http://hea-www.cfa.harvard.edu/~arots/nvometa/STC/STCunits.txt` Unit strings allowed in STC (Rots)

- `http://www.mel.nist.gov/msid/sima/07_ndml.htm`
  NIST (National Institute of Standards & Technology) project UnitsXML builds up an XML representation of units at the granularity level of a simple symbol string
- FITS standards:`http://fits.gsfc.nasa.gov/fits_wcs.html`
  `http://fits.gsfc.nasa.gov/fits_draft.html`
  Reference, Development and Specification of Physical Units within OGIP FITS files (Greisen and Calabretta 2002, A&A 395, 1061)
  `http://heasarc.gsfc.nasa.gov/docs/heasarc/ofwg/docs/general/ogip_93_001/`
- `https://jsr-275.dev.java.net/`
  JAVA JSR-275 specifies Java packages for the programmatic handling of physical quantities and their expression as numbers of units.
- `aips++` `http://aips2.nrao.edu/docs/aips++.html` and
  `casacore` `http://code.google.com/p/casacore/`
  contain modules handling units and quantities with high precision. The packages are mainly in use for radio astronomy but are designed to be modular and adaptable. (NB contrary to the statement on the casacore link, aips++ is still very much in use as the toolkit behind the CASA package.)
- IAU SOFA `http://www.iau-sofa.rl.ac.uk/` and
  USNO NOVAS `http://aa.usno.navy.mil/software/novas/novas_info.php`
  implement the IAU 2000 recommendations.

3.2 Quantities

A quantity, e.g. a measurement of a physical value like the speed of light, has a value (2.998 10+5), a ucd (phys.veloc), units (km.s-1) and is coded using a numerical type (real).

Some quantities are also reused as units. Many units are expressed, or converted, in terms of physical constants such as the speed of light,

- $c =$ `2.998 10+8 m.s-1;`
- Boltzman's constant, K_B =`1.38065 10-23`
- `1 AU` = `1.499 10+11 m.`

Many of these are used as units in their own right, e.g. velocities may be expressed as a fraction or multiple of c, but c is also used to convert between wavelength and frequency, etc. These are combinations of units with scaling factors applied, and so can be treated in the same way as any other compound unit e.g. the `Jy` (`10-26 W.m-2.Hz-1`) .

We need to ensure that we are consistent with the IVOA Quantity model, where appropriate.

3.3 Libraries

Some of the links in Section 3.1 Vocabularies, e.g. for Java, contain libraries. One of the most widely-used specialised astronomical libraries is AST which includes a unit conversion facility attached to astronomical coordinate systems. In all cases, some software effort is required if they are to be used in translating between data provider (etc.) unit labels, and those to be adopted by the IVOA for internal use. Work is ongoing by *David Berry* to make AST compatible with **STC**, the IVOA standard for coordinates.

4 Proposed IVOA conventions

4.1 Unit labels

These are standards for the vocabulary and syntax which we suggest are used internally within the VO. We need to provide translation to/from other usages as far as is required/practical.
Please supply amendments and additional proposals for this section

4.1.1 First WG proposal

1. We pick one label system from Section 3.1 – STC – as the basis, to be expanded where required.
2. We use SI internally (not cgs etc.)
3. We recognise European or US spelling (`metre` or `meter`)
4. `m` as wavelength is distinguished from `m` as length by providing different UCDs in the physical quantity using this unit label
5. SI prefixes directly precede a unit, no space should be used, e.g. `mT` = milli-Tesla and `Tm` = Tera-metre
6. Compound units are formed using the dot character '.' as a separator, without spaces, e.g. `N.m` = Newton metre *This helps to avoid ambiguity but AST uses a different convention, N m - however, spaces are ignored by some parsers.*
    - In addition, a delimiter may be needed to indicate a unit label e.g. `[N.m]` in order to avoid ambiguity with Utypes and table aliases in ADQL, which are also indicated e.g. *table.columnname*
    - Specific cases should be considered: beware e.g. `MJD` which is not Mega Julian Day but `Modified Julian Day`. Therefore, interpreters will have to first check whether the unit label matches a known unit, and only if not, strip off the first character, check if

that matches an SI prefix and the remainder a unit string, and so on.

7. Powers are indicated by + or - e.g. `m+2` = `metre`× `metre`, `m.s-1` = `m/s`

    - Many existing libraries do not allow fractional powers (nor do they allow e.g. logarithms). These might be required by users applying algorithms to derive physical quantities e.g. synchrotron radiation intensity is proportional to the 2/7 power of the magnetic field strength (optical depth is a function of the natural logarithm of flux density).
    Can we include these?

4.1.2 Proposal based on FITS

http://heasarc.gsfc.nasa.gov/docs/heasarc/ofwg/docs/general/ogip_93_001/ described the construction of units in FITS. This has two parts. The list of allowed base units might have to be extended to cover the range of units which need to be *recognised*, and possibly those which are used internally. The rules for constructing compound units might be suitable, or at least serve as a basis. Possible issues include:

- Space is the recommended multiplicative separator, although * is permitted.
- Round brackets () are used; we may need a different style to avoid clashes?

Note that this discussion refers to the rules for parsing and constructing unit strings inside the VO; we may need to be able to *recognise* a wider range of labels and syntax in the translation layer which examines data as published.

4.2 Unit operations

Conversion between recognised units, including factorising into SI multiples, e.g.
`100 J` × `10 s-1` = `1000 J.s-1` = `1000 W` = `1 kW`
can be accomplished by tools provided by the IVOA, based, as far as possible, on existing libraries and techniques such as dimensional analysis.

Some unit conversions (including changes of origin between labels in the same basic unit), e.g. `0 degC` = `273 K`, may also be included conveniently in such packages, if they do not require external information or mathematical operations which are excessively complicated for the environment. Section 2.1 gives examples of operations outside the scope of this package and any implementation should make these clear and provide pointers to alternative services where necessary, if possible.

5 Use cases for the IVOA Units model

5.1 Simple use cases

- Using SI multiplicative prefixes: a result is returned containing values like `12345 GHz` or `0.0003 MHz`; these should be listed, or axes of plots labelled, with respect to `12.345 THz` or `0.3 kHz`.
- Linear unit conversion: an image has a pixel scale in degrees e.g. `0.0002 deg`; the display in Aladin should provide information in `arcsec`
- Handling unit relationships and compound units: a query involves deriving the flux for a given photon emission rate (in `W`) from Planck's constant (`6.6` 10^{-34} `J.s`), the radiation frequency (in `GHz`), and the number of photons emitted per second.
- Handling astronomical constants requiring specific unit forms, e.g. cosmological distance, in `Megaparcsec`, $= c.z.H$`0-1`
  where `c` is in `km.s-1`
  z (redshift) is (lambda_obs - lambda_rest)/lambda_rest, i.e. the ratio of `m-m/m`, or dimensionless
  and the Hubble constant H0 is in `km.s-1.Mpc-1`.

5.2 Overview of units use in the VO framework

Different VO entities require and consume metadata with units attached like registries, applications and interoperate via protocols. Figure 1 illustrates the places where the IVOA could intervene to ensure consistent use of units.

5.3 ADQL use

These suggestions arise from consultation with Jeff Lusted of the ADQL group. The following proposals are of increasing complexity and need not all be implemented at once; 1. is the starting point and 2. follows.

1. We should encourage all data providers to supply units for each column of a table. In most published tables in Astronomical journals and Vizier

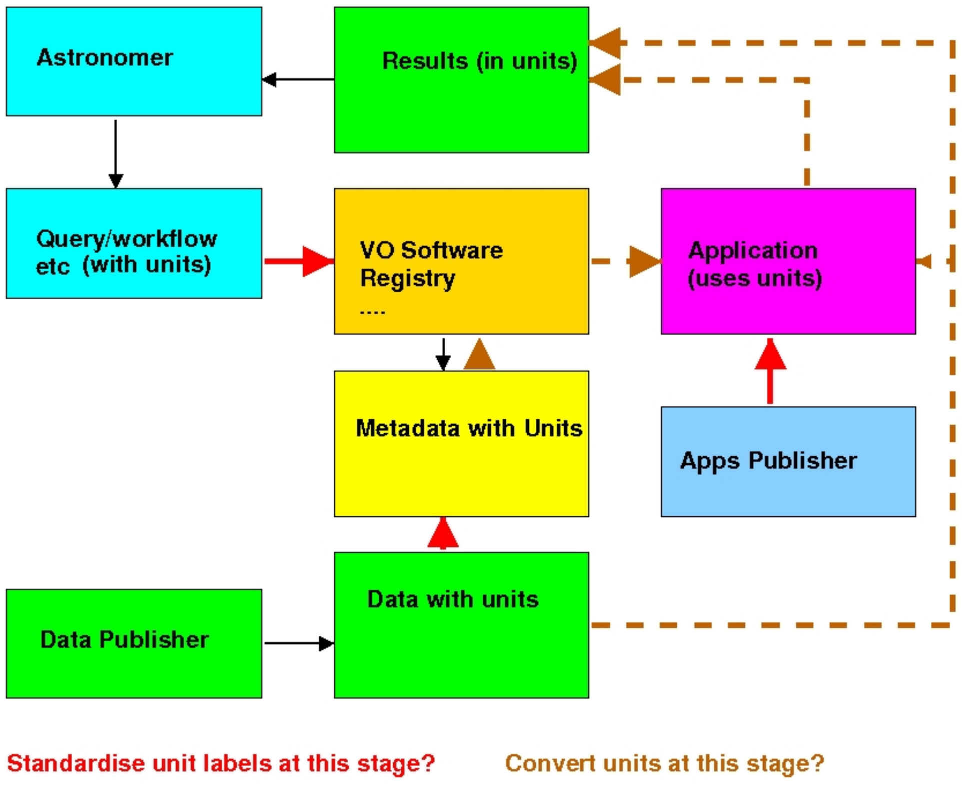
Astronomer
Results (in units)
Query/workflow etc (with units)
VO Software Registry ....
Application (uses units)
Metadata with Units
Apps Publisher
Data Publisher
Data with units
Standardise unit labels at this stage?
Convert units at this stage?

Figure 1: *This shows the levels at which conversions might be done. Thick red arrows: At the point where an astronomer or data provider submits input to the VO, we should provide tools to ensure that units are labeled consistently according to the labelling convention adopted. This implies that a units parsing step is included prior to metadata ingestion into the VO. Dashed brown arrows: Conversions required to supply results to the user in specified or reasonable units e.g.* `J.s-1` *to* `W`*, are done where and when they are required.*

server as well, unitless values are represented by "—". This could be adopted for the VO convention as well. The IVOA needs to provide a parser – *inside the ADQL standard?* to relate the native units to the standard IVOA labels. Columns should also have associated UCDs.

2. The default response to a query which does not specify units, will be in the native units of the table. *We recommand that the output units will be labelled using the IVOA standard label ???*
3. Where queries involve combining or otherwise operating on the content of columns to produce an output column with modified units, we can provide libraries and a parser to assist in assigning and checking a new unit, and attach this to the returned values via the SQL CAST operator. This is implemented already in database related applications such as Saada, for instance. *Which group should implement this?* If any column used in responding to a query lacks a necessary unit, the output involving that column will be unitless.
4. If the user wants to change the output units with respect to the table units, this could be done by specifying the units in the initial SELECT statement. There are several issues to consider:
   (a) Does the user also need to include the conversion expression, or does the unit parser take care of that?
   (b) Can the user use this to assign units (based on prior knowledge) to output from a column lacking a unit?

5.4 Model representations

See also Section 3.3, in particular **AST**, which provides tools for mapping between units, converting multiples of SI prefixes, and simple coordinate conversion.

5.4.1 A grammar for unit strings

to be completed and stabilized

The precise form will depend upong the syntax adopted.

Backus Naur format (as used e.g. for ADQL) provides cross-references to define terms, e.g.
Jy = 10-26 W m-2 Hz-1
W = J.s-1
...
Jy = 10-26 kg.m+2.s-2.s-1.m-2.s

here we provide a recursive (BNF) definition of a Unit string representation as follows:

```
Full_Unit = factor Complex_unit
 Complex_Unit = single_Unit
                | single_Unit OP Complex_Unit
                | "log[" single_Unit OP single_Unit "]"
                | "mag[" single_Unit OP single_Unit "]"
 single_Unit  = extended_UnitSymbol
                | extended_UnitSymbol power
 extended_UnitSymbol = UnitSymbol
                       | Magnitude_Prefix UnitSymbol
                       | "(" Full_Unit ")"
 power = Number
         | +Number
         | -Number
 OP      = . | /
 factor=?
 Magnitude_Prefix UnitSymbol=?
```

5.4.2 Dimensional Analysis

The dimensional analysis system described in
http://arxiv.org/pdf/astro-ph/0511616 allows to fully define and describe the unit concept and build up conversions methods from one string representation to another one.
Are the library and tools used available outside VOSpec, for incorporation in other tools?

5.4.3 UML model

We present Fig. 2 as an example to help visualisation, since the hope of reusing existing libraries implies that there is no need to generate an exhaustive representation of every possible unit in this form. The model aims at clarifying the role of Units and Quantities and defines their properties and dependencies.

This simple Unit Model is made of only two classes, the ***Units*** class and the ***Quantity*** class which gathers the relevant information to describe a physical quantity used in any observational science like astronomy. This class illustrates the concept of a physical value with its attached metadata. Most of VO applications use or will indeed use their own derivation of this

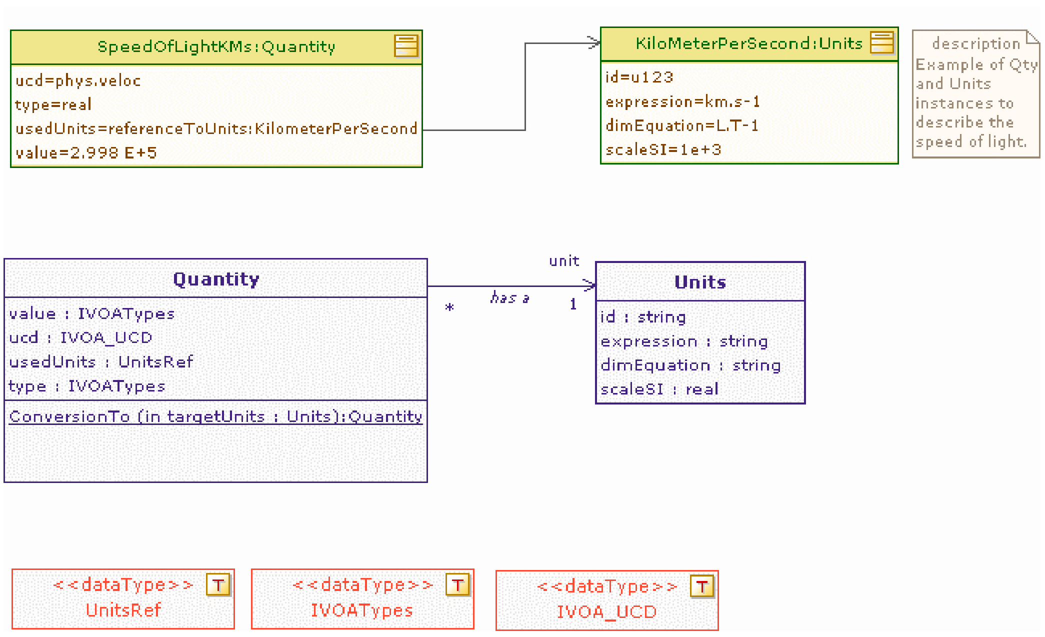

Figure 2: *UML model for units.* Bottom: *Simple types defined by the model and used for class attributes.* Middle: *This simple class diagram illustrates how the Quantity Class might use the Units Class to represent which unit string is used to code the value attribute.* Top: *As an example, a specific Quantity, SpeedOfLight, is represented by a Quantity instance object and uses the KilometerPerSecond object, an instance of the Units Class. Such a Units instance could be used by many Quantity objects.*

concept. A ***Quantity*** contains a **value** attribute, to store a numerical value of a specified type (real, integer, ..., IVOA type), a **typename** for this value, a **reference** to a ***Units*** class instance, and a **UCD string**, as defined in the IVOA standard document for Unified Content Descriptors
(`http://www.ivoa.net/Documents/latest/UCD.html`
and `http://www.ivoa.net/Documents/cover/UCDlist-20070402.html`.
if two Quantity objects Q1 , Q2 point to the same Units class *Meter*, with expression '`m`' , their ucd attribute will help to disentangle wavelength from distance, with for instance Q1.ucd='em.wl', and Q2.ucd='pos.dist'. Quantity Units conversions are needed to modify the value of input or output parameters for applications or protocols in the VO ; they are applied to a ***Quantity*** instance, and therefore are defined as methods of the ***Quantity*** class. The ***Units*** Class describes the details necessary to handle units in a general way: it is represented by a string expression built up on SI symbols as defined in the IAU units list or Vizier supported units symbols: this corresponds to the **expression** attribute of the ***Units*** class. It can have an **id**, a **name**; it is mathematically described by an equation reflecting which physical variables correspond to this unit as refered above and developed in Osuna & Salgado (2008). For instance, the dimensional Equation of `Jy` is '1.E-23 MT-2'. In this Model, **scaleSI** is the Scale factor, here 1.E-23 and `MT-2` the **dimEquation** attribute coded as a string.

This ***Units*** class is simple on purpose, and can be derived for various use-cases, for example to add a unit class name, a boolean : "SI or not SI", etc... The units conversion deals with the ***Quantity*** Class, and aims at transforming one ***Quantity*** instance of a specified (value, unit) pair , e.g. Q1(100, KilometerPerSecond) pointing to the ***Units*** instance class *KilometerPerSecond* of expression `"km.s-1"` into another ***Quantity*** instance Q2(10 E+5, MeterPerSecond) which points to the "*MeterPerSecond*" Units object whose expression attribute contains `"m.s-1"`.
The conversion methods of the ***Quantity*** class can implement the AST library or units conversion methods defined in SPLAT, for instance.

6 Implementation of the IVOA Units model

The astronomical community has been dealing with units as strings for a long time, usually effectively. This model does not seek to redefine or forbid any particular usage of unit strings, but to provide guidelines for services that want/need to convert between units. The grammar identifying the unit strings which are recognised and encouraged in the Virtual Observatory can be implemented in order to parse the unit strings as they are already used in various serialisations: XML instance document or VOTable. Examples are provided in appendices A and B.

References

[AST] AST Library, D. Berry at al. 2009,
http://www.starlink.ac.uk/~dsb/ast/sun211.htx/sun211.html

[SLALIB] http.www.starlink.rl.ac.uk/star/docs/sun67.htx/sun67.html

[Osuna P. & Salgado, J., 2008] http://arxiv.org/pdf/astro-ph/0511616

A Appendix A: Units inside XML serialisations : an XMM observation

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Char level 4 example : on Xray source  2XMM J001708.8+813508 -->
<!-- describes  the spectral response of an observed RGS spectrum and how to ace
‘‘ format file -->
<!-- author Mireille Louys/ Laurent Michel -->
<!-- edited with Oxygen -->

<characterisation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ivoa.net/xml/Characterisation/Characterisation-v1
 xmlns="http://www.ivoa.net/xml/Characterisation/Characterisation-v1.11.xsd"
 xmlns:stc="http://www.ivoa.net/xml/STC/stc-v1.30.xsd "
 xmlns:xlink="http://www.w3.org/1999/xlink">
<characterisationAxis><!-- SPATIAL-->
<axisName>spatial</axisName>
<ucd>pos</ucd>
<unit>deg</unit>
<coordsystem id="TT-ICRS-TOPO" xlink:href="ivo://STClib/CoordSys#TT-ICRS-TOPO" x
<independentAxis>true</independentAxis>
<calibrationStatus>CALIBRATED</calibrationStatus>
<numBins2>
<I1>1</I1>
<I2>1</I2>
</numBins2>
<undersamplingStatus>true</undersamplingStatus>
<regularsamplingStatus>true</regularsamplingStatus>
          <coverage>
         <location>
         <coord coord_system_id="TT-ICRS-TOPO">
<stc:Position2D>
<stc:Name1>RA</stc:Name1>
<stc:Name2>Dec</stc:Name2>
<stc:Value2>
<stc:C1>8.3344E-02</stc:C1><!--d’apres le header   -->
<stc:C2>-3.2350E+01</stc:C2><!-- +81:35:08.7   -->
</stc:Value2>
</stc:Position2D>
         </coord>
         </location>
          </coverage>
        </characterisationAxis>
<characterisationAxis>
<axisName>time</axisName>
<ucd>time</ucd>
<unit> none</unit>
<!-- none unit is for ISO-8601 format -->
<coordsystem idref="TT-ICRS-TOPO"/>
<independentAxis>true</independentAxis>
<calibrationStatus>UNCALIBRATED</calibrationStatus>
<numBins1>1</numBins1>
          <coverage>
         <location>
         <coord coord_system_id="TT-ICRS-TOPO">
<stc:Time>
<stc:TimeInstant>
<stc:ISOTime>22001-08-23T03:09:16.29</stc:ISOTime>
</stc:TimeInstant>
</stc:Time>
          </coord>
          </location>
          <bounds>
          <limits>
          <CoordArea  coord_system_id="TT-TOPO-ICRS">
          <stc:TimeInterval>
          <stc:StartTime>
          <ISOTime>2001-08-23T03:31:49</ISOTime>
          </stc:StartTime>
          <stc:StopTime>
          <ISOTime>2001-08-23T15:08:43</ISOTime>
          </stc:StopTime>
          </stc:TimeInterval>
          </CoordArea>
          </limits>
          </bounds>
         </coverage>
</characterisationAxis>
<characterisationAxis>
<axisName>EnergyChannelNb</axisName>
<ucd>phys.energy</ucd>
```

```
<unit>none</unit>
<coordsystem idref="TT-ICRS-TOPO"/>
<independentAxis>true</independentAxis>
<calibrationStatus>CALIBRATED</calibrationStatus>
<numBins1>3400</numBins1>
<undersamplingStatus>false</undersamplingStatus>
<regularsamplingStatus>false</regularsamplingStatus>
        <coverage>
         <location>
         <Interval> </Interval>
         </location>
            <bounds>
        <limits coord_system_id="TT-ICRS-TOPO">
<CoordScalarInterval>
<stc:LoLimit>   3.56 </stc:LoLimit>
<stc:HiLimit>   4.00  </stc:HiLimit>
</CoordScalarInterval>
        </limits>
       </bounds>
        </coverage>
        <resolution>
         <unit> km/s </unit>
        <resolutionRefVal>
<stc:Resolution>1.00 </stc:Resolution>
        </resolutionRefVal>
        <resolutionBounds>
<resolutionLimits1>
<stc:LoLimit>   0.7597   </stc:LoLimit>
<stc:HiLimit>     1.161    </stc:HiLimit>
</resolutionLimits1>
      </resolutionBounds>
                </resolution>
</characterisationAxis>
<!-- update for Xmm spectrum  -->
<characterisationAxis>
<axisName>flux</axisName>
<ucd>counts</ucd>
<unit>none</unit>
<coordsystem id="UNKNOWN"/>
<accuracy>
<statError>
```

```
<flavor>statistical</flavor>
<ErrorRefVal>
<stc:Error>1.00e-17</stc:Error>
</ErrorRefVal>
<ErrorBounds>
<ErrorLimits1>
   <stc:LoLimit> 0.7e-17 </stc:LoLimit>
   <stc:HiLimit>  1.7e-17</stc:HiLimit>
</ErrorLimits1>
</ErrorBounds>
          </statError>
    </accuracy>
<independentAxis>false</independentAxis>
<calibrationStatus>UNCALIBRATED</calibrationStatus>
<numBins1>0</numBins1>
<undersamplingStatus>false</undersamplingStatus>
<regularsamplingStatus>true</regularsamplingStatus>
 <coverage>
       <location>
<coord coord_system_id="UNKNOWN">
<stc:ScalarCoordinate>
<stc:Value> 10.0e-17 </stc:Value>
</stc:ScalarCoordinate>
             </coord>
        </location>
                      <bounds>
         <limits coord_system_id="UNKNOWN">
<CoordScalarInterval>
<stc:LoLimit> 1.3e-17 </stc:LoLimit>
<stc:HiLimit> 24.0e-17</stc:HiLimit>
</CoordScalarInterval>
        </limits>
        </bounds>
 <sensitivity>
<variationMap>
                    <Map>
                         <type>table</type>
                         <format>table/fits</format><!--mimetype-->
                         <datamodel>proprietary</datamodel><!--link to XML schem
                         <acref> http://xcatdb.u-strasbg.fr/2xmmi/getinstance?oi
                         <Access>
```

```
                          <AccessParams>
                                <!--  <Extname>SPECTRUM</Extname> -->
                                <field>Error</field>
                                <unit>1/s cm^2 Angstrom</unit>
                             <size>100 </size>
                            </AccessParams>
                         </Access>
</Map>
                    <documentation> documentation on error map.</documentation>
        </variationMap>
     </sensitivity>
 </coverage>
</characterisationAxis>
</characterisation>
```

B Appendix B: Units in VOTable serialisation: a Spectrum example

C Appendix C: Updates of this document

- version 0.1 to 0.2
  - 20090521
    - added UCD to Quantity in point 4 of subsection 4.1
    - added '.' in the notation in unit strings in section 5.1
    - added a sentence on the help of UCd in quantity in section 5.4.3
  - 20090522
    - clarified the scope of the model in Section 2.1
    - added references in Section 3.1
    - added requirement to be consistent with Quantity DM in Section 3.2
    - minor clarification and subediting