International
Virtual
Observatory
Alliance

Units in the VO
Version 1.0

IVOA Proposed Recommendation 2013-07-01

This version:

http://www.ivoa.net/Documents/VOUnits/20130701/

Latest version:

http://www.ivoa.net/Documents/VOUnits/

Previous versions:

http://www.ivoa.net/Documents/VOUnits/20130419/

Editor(s):

Sébastien Derrière

Authors:

Sébastien Derrière, Norman Gray, Mireille Louys, Jonathan McDowell, François Ochsenbein, Pedro Osuna, Anita Richards, Bruno Rino, Jesus Salgado

Contents

Abstract

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

The core of the document contains the recommended rules for writing string representations of unit labels, called VOUnits.

Status of this document

This is an IVOA Proposed Recommendation made available for public review. It is appropriate to reference this document only as a recommended standard that is under review and which may be changed before it is accepted as a full recommendation.

This document is a substantial update of the previous version 0.2 that was written within the Data Model IVOA Working Group. As decided in previous IVOA interoperability meetings, the Semantics working group is now in charge of the document. This document is intended to become a full IVOA recommendation, following agreement within the community and standard IVOA recommendation process.

The place for discussions related to this document is the Semantics IVOA mailing list `semantics@ivoa.net`.

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 providing comments, especially Markus Demleitner, Paddy Leahy, Jeff Lusted, Arnold Rots, Mark Taylor, Brian Thomas and recent contributors on the DM forum.

1 Introduction

This document describes a standardised use of units in the VO (hereafter simply "VOUnits"). It aims to describe a syntax for unit strings which is in the intersection of existing syntaxes, and to list a set of 'known units' which is the union of the 'known units' of those standards. This aim is not quite possible in fact, and the mild deviations from it are discussed below in Sect. 2 and Appx. C; there is a summary of the various units in Table 1 on page 8. We *recommend*, therefore, that applications which write out units should do so using *only* the VOUnits syntax, and that applications reading

units should be able to read *at least* the VOUnits syntax, plus all of the units of Sect. 2.2.

The introduction gives the motivation for this proposal in the context of the VO architecture, from the legacy metadata available in the resource layer, to the requirements of the various VO protocols and standards and applications.

This document is organised as follows: Sect. 2 details the proposal for VOUnits. Sect. 3 lists some use cases and reference implementations. In Appx. A, there is a brief review of current practices in the description and usage of units; in Appx. B there is a detailed discussion of the differences between the various syntaxes; and in Appx. C there are formal (yacc-style) grammars of the four syntaxes discussed.

The normative content of this document is Sect. 2 and Appx. C.4.

1.1 Units in the VO Architecture

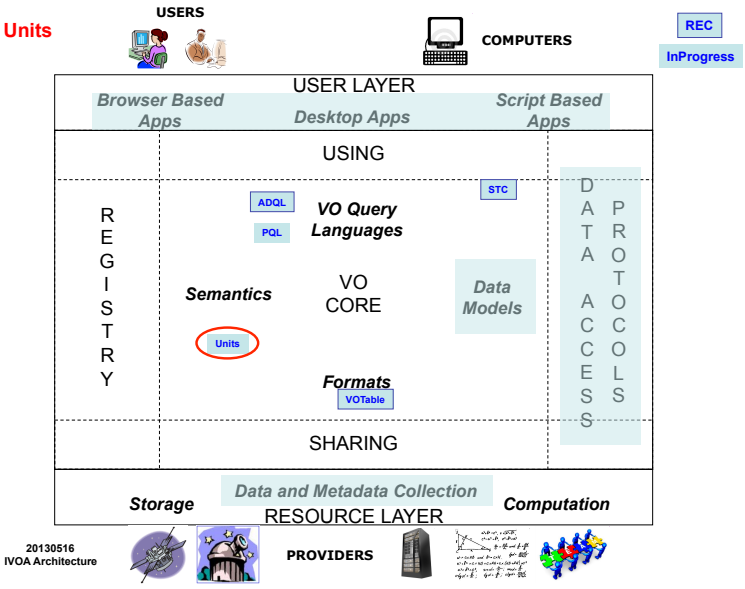

Figure 1: Units is a core building block in the VO. Most parts of the architecture rely on it: the User Layer with tools and clients, the Resource Layer with data. Protocols and registries entries, as well as data models re-use these Units definitions.

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

The VOUnits lie at the core of the VO architecture, as can be seen in

Fig. 1. This VO standard has no dependency on any VO standards but rely on former units string definitions like the FITS convention for instance. On the contrary, this specification is implicitely reused in many other VO standards definitions.

Most of the existing data and metadata collections accessible in the resource layer do have some legacy units, which are mandatory for any scientific use of the corresponding data. Units can be embedded in data (e.g. FITS headers) or be implied by convention and/or (preferably) specified in metadata.

Units also appear in the VOTable format (Ochsenbein et al., 2011), through the use of a `unit` attribute that can be used in the `FIELD`, `PARAM` and `INFO` elements. Because of the widespread dependency of many other VO standards on VOTable, these standards inherit a dependency on Units.

The Units also appear in many Data Models, through the use of dedicated elements in the models and schemas. At present, each VO standard either refers to some external reference document, or provides explicit examples of the Units to be used in its scope, on a case-by-case basis.

The registry records can also contain units, for the description of table metadata. The definition of VO Data Access protocols uses units by specifying in which units the input parameters have to be expressed, or by restricting the possible units in which some output must be returned.

And last but not least, tools can interpret units, for example to display heterogeneous data in a single diagram by applying conversions to a reference unit on each axis.

1.2 Adopted terms and notations

Discussions about units often suffer from misunderstandings arising from cultural differences or ambiguities in the adopted vocabulary. For the sake of clarity, in this document, the following concepts are used:

- a **quantity** is the combination of a (numerical) *value*, measured for a *concept* and expressed in a given *unit*. In the VO context, the nature of the concept can be expressed with a UCD or a utype. This document does not address the full issue of representing quantities, but focusses on the *unit* part.
- a **unit** can be expressed in various forms: in natural language (e.g. *metres per second squared*), with a combination of symbols with typographic conventions (e.g. $\mathrm{m\,s^{-2}}$), or by a simplified text label (e.g. m.s-2). VOUnit deals with the label form, which is easier to standardize, parse and exchange. A VOUnit corresponds in the most general case to a combination of several (possibly prefixed) symbols with mathematical operations expressed in a controlled syntax.

- a **base unit** is represented by a **base symbol**, with unambiguous meaning.
- a **prefix** or **scale factor** is prepended to a **base symbol** to scale it by factors of ten.
- a **symbol** or sym is either a base symbol or a prefixed base symbol.

Remark: some complex questions, more related to data modeling than to units, such as how a quantity is associated to its measurement error, or how groups of coordinates are described, are not addressed in this document. They can always be broken down, with appropriate modeling, into smaller bits to which VOUnits can be applied.

1.3 Purpose of this document

The purpose of this document is to provide a reference specification on how to write VOUnits, in order to maximize interoperability within the VO.

VOUnits will not try to reinvent the wheel, and will be as compliant as possible with legacy metadata in major archives, and astronomers' habits.

By explicitly stating what must (and must not) be used, what may be used, and giving clear recommendations as to the preferred practices, it should avoid some common misinterpretations and confusions that are often encountered when dealing with units. In particular:

- We describe (Appx. A) a number of existing unit syntaxes, and mention some ambiguities in their definition. Application authors should expect to encounter each of the syntaxes mentioned in this document (FITS, OGIP and CDS); all of these are broadly endorsed by this specification.
- Where there are some ambiguities in, or contradictions between, these specifications, we recommend that application authors should resolve them as indicated in this specification.
- This document defines a syntax, called 'VOUnits', which is as far as is feasible in the intersection of the three syntaxes, and which we recommend that applications should use when writing unit strings.

1.4 What this document will not do

This document is **not** prescribing what units data providers employ, nor enforcing that a given quantity be expressed in a unique way (e.g. all distances in m). So long as data are labelled in a recognised system, a translation layer can 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 his or her query and for the result.

VOUnits do not specify how transformation of quantities is done. But VOUnits describe the units of each quantity in a standard way, allowing to do the conversion if the underlying transformation model is known. This

is the case for simple operations such as converting light wavelengths into frequencies, but also for more complex ones such as coordinate conversions or converting magnitudes into fluxes.

This specification describes only isolated units, and not arrays, records or other combinations of units. Thus, it does not cover declaring an RA-Dec pair, for example. Several VO protocols require embedding complex objects into result table and give string serializations for those: geometries in TAP results are the most common example. This specification does not cover this situation, although we hope that where individual unit strings are covered in such instances, their syntax will conform to this specification.

2 Proposal for VOUnits

By systematically comparing the four reference schemes described above (details are given in Appendix B), the rules for VOUnits are defined in this section. Various aspects are addressed:

- how the labels are encoded;
- what base symbols are allowed and how they are spelled;
- what prefixes are allowed and how they are used;
- how symbols are combined.

A formal grammar summarizing these conventions is given in Appx. C.4.

The text below is expected to be compatible with the prescriptions of BIPM (2006).

2.1 String representation and encoding

VOUnits must be unit labels suitable for IVOA use (or other electronic manipulation), and therefore need to be expressed unambiguously without encountering encoding problems.

Following the current usage in unit labels (see Table 8 on page 19), *VOUnits must be case-sensitive strings of chars consisting of printable ASCII characters* (values coded 20 to 7E in hexadecimal). The syntax therefore excludes special characters such as Å or µ.

2.2 Known units

In Table 1 on the following page, we indicate the 'known units' for each of the described syntaxes. In the table, a '·' indicates that the unit is recognised in that syntax, an 's' that it it additionally permitted to have SI prefixes, and a 'd' that it is recognised but deprecated. There are a few units (namely '**angstrom** or **Angstrom**', **pix** or **pixel**', '**ph** or **photon**' and '**a** or **yr**') for which there are recognised alternatives in some syntaxes, and in these cases 'p' marks the preferred one.

unit	*fits*	*ogip*	*cds*	*vou*	*unit*	*fits*	*ogip*	*cds*	*vou*
%			·		lm	s	s	s	s
A	s	s	s	s	lx	s	s	s	s
a	s		s	s	lyr	·	·		s
adu	·			s	m	s	s	s	s
Angstrom	d		·	dp	mag	s	·	s	s
angstrom		·		d	mas	·		·	
arcmin	·	·	·	s	min	·	·	·	s
arcsec	·	·	s	s	mol	s	s	s	s
AU	·	·	·	s	N	s	s	s	s
barn	sd	·	s	sd	Ohm	s		s	s
beam	·			s	ohm		s		
bin	·	·		s	Pa	s	s	s	s
bit	s		s	s	pc	s	s	s	s
byte	s	·	s	s	ph	·			s
C	s	s	s	s	photon	p	·		sp
cd	s	s	s	s	pix	·		·	s
chan	·	·		s	pixel	p	·		sp
count	·	·		sp	R	s			s
Crab		s			rad	s	s	s	s
ct	·		·	s	Ry	·		s	s
d	·	·	·	s	s	s	s	s	s
D	·		·	s	S	s	s	s	s
deg	·	·	·	s	solLum	·		·	s
erg	d	·		sd	solMass	·		·	s
eV	s	s	s	s	solRad	·		·	s
F	s	s	s	s	sr	s	s	s	s
g	s	s	s	s	T	s	s	s	s
G	sd	·		sd	u	·			s
H	s	s	s	s	V	s	s	s	s
h	·	·	·	s	voxel	·	·		s
Hz	s	s	s	s	W	s	s	s	s
J	s	s	s	s	Wb	s	s	s	s
Jy	s	s	s	s	yr	sp	·	sp	sp
K	s	s	s	s					

Table 1: Known units in the various syntaxes

We recommend that unrecognised units be accepted, as long as they are parsed giving preference to the syntaxes and prefixes described here. Thus the string furlong/week should parse successfully (though perhaps with suitably-communicated warnings) as the femto-'urlong' per week.

2.3 Base units

There is good agreement for the base symbols across the different schemes (see Table 9 on page 19). VOUnits follow the same notations.

The VOUnits base symbols are listed in Table 2

m (metre)	g (gramme)	J (joule)	Wb (weber)
s (second of time)	rad (radian)	W (watt)	T (tesla)
A (ampere)	sr (steradian)	C (coulomb)	H (henry)
K (kelvin)	Hz (hertz)	V (volt)	lm (lumen)
mol (mole)	N (newton)	S (siemens)	lx (lux)
cd (candela)	Pa (pascal)	F (farad)	Ohm (ohm)

Table 2: VOUnits base units

For masses, the SI unit is kg. However, existing specifications recommend not to use scale factors with kg, but with g if needed, which makes the parsing potentially difficult. In VOUnits, only the g base symbol is defined, and kg is implicitly allowed by using the 'k' scale factor with this base symbol.

This is just a way to simplify the parsing, by simultaneously allowing scale factors for g and disallowing additional scale factors for kg.

No use of the Ohm was found in astronomy archives, therefore the possible non-compliance of the OGIP syntax was not considered critical.

Recognising a known unit takes priority over parsing for prefixes. Thus the string Pa represents the Pascal, and not the peta-year, and the string mol will always be the mole, and never a milli-'ol', for some unknown unit 'ol'.

2.4 Scale factors

Following the usage in Table 10 on page 20, there are 20 scale factors that can be used as prefixes to any base symbol (except that P is not allowed as a scale factor for a)

These scale factors – provided in Table 3 – are the same as those of BIPM (2006) and of Table 5 of Pence et al. (2010).

One must not use compound prefixes (that is, more than one SI prefix). Prefixes are concatenated to the base symbol without space, and cannot be used without a base symbol.

Remark: the letter u is used instead of the μ symbol to represent a factor of 10^{-6}, following the character set defined in Sect. 2.1.

d (deci – 10^{-1})	p (pico – 10^{-12})	da (deca – 10^{1})	T (tera – 10^{12})
c (centi – 10^{-2})	f (femto – 10^{-15})	h (hecto – 10^{2})	P (peta – 10^{15})
m (milli – 10^{-3})	a (atto – 10^{-18})	k (kilo – 10^{3})	E (exa – 10^{18})
u (micro – 10^{-6})	z (zepto – 10^{-21})	M (mega – 10^{6})	Z (zetta – 10^{21})
n (nano – 10^{-9})	y (yocto – 10^{-24})	G (giga – 10^{9})	Y (yotta – 10^{24})

Table 3: VOUnits prefixes

The SI prefixes of Table 10 on page 20 always refer to multiples of 1000, even when applied to binary units such as bit or byte. This Recommendation does not discuss the binary prefixes (kibi, Mibi, and so on) of IEC 80000-13, §4.

2.5 Astronomy symbols

Table 11 on page 21 lists symbols used in astronomy to describe times, angles, distances and a few additional quantities. The subset of these used by this specification are listed in Table 4.

min (minute of time)	deg (degree of angle)	AU (astronomical unit)
h (hour of time)	arcmin (arcminute)	pc (parsec)
d (day)	arcsec (arcsecond)	eV (electron volt)
a or yr (year)	mas (milliarcsecond)	Jy (jansky)
u (atomic mass)		

Table 4: Additional astronomy symbols

Minutes, hours, and days of time must be represented in VOUnits by the symbols min, h and d; however the cd is the candela, not the centi-day. The year can be expressed by a (recommended by IAU) or yr (common practice), but peta-year must only be written Pyr, to avoid the collision with the pascal, Pa.

The astronomical unit must be expressed in upper-case, AU, in order to follow the legacy practice, to avoid the (admittedly unlikely) confusion with atto-atomic mass, and for the sake of consistency.

There are no VOUnit symbols for celsius degrees or century. Temperatures are expressed in kelvin (K), and a century corresponds to ha or hyr. Note that the unit ha is *not* hectare, which is the SI stipulation (BIPM, 2006).[1]

A few symbols which might be ambiguous are listed in Table 5, with their correct VOUnit interpretation and what they do not mean.

[1] If large telescope arrays wish to talk of Watts per Hertz per hectare, for some reason, they're going to have to find some other way of doing so.

VOUnit	Correct interpretation	Incorrect
Pa	pascal	peta-year
ha	hecto-year	hectare
cd	candela	centi-day
au	atto-atomic-mass	astronomical unit

Table 5: Possibly ambiguous units

2.6 Other symbols

Table 12 on page 22 corresponds to Table 7 in the IAU document, and the IAU strongly recommends no longer using these units. Data producers are strongly advised to prefer the equivalent notation using symbols and prefixes listed in Tables 9, 10 and 11.

However, in order to be compatible with legacy metadata, VOUnit parsers should be able to interpret symbols for ångström, barn, erg and gauss, as below:

angstrom or Angstrom	erg
barn	G (Gauss)

Table 13 on page 23 compares other miscellaneous symbols.

The last set of VOUnits symbols, derived from this comparison, is in table Table 6

mag (magnitude)	pix or pixel	solMass (solar mass)	R (rayleigh)
Ry (rydberg)	voxel	solLum (solar luminosity)	chan (channel)
lyr (light year)	bit	solRad (solar radius)	bin
ct or count	byte (8 bits)	Sun (relative to the Sun, e.g. abundances)	beam
ph or photon	adu	D (Debye)	? (unknown)

Table 6: Miscellaneous VOUnits

It can be noted that some of the units listed in Table 13 on page 23 are questionable. They arise in fact from a need to describe quantities, when the only piece of metadata available is the unit label. Count, photon, pixel, bin, voxel, bit, byte are concepts, just as apple or banana. The associated quantities could be fully described with a UCD, a value and a void unit label.

It is possible to count a number of bananas, or to express a distance measured in bananas, but this does not make a banana a reference unit.

The FITS document provides the most general description of all the compared schemes, and VOUnits adopts similar definitions, in order to acknowledge at best legacy metadata. The VOUnits symbol for magnitudes is **mag**. The symbol **Sun** is used to express ratios relative to solar values, for example abundances or metallicities. Note that all symbols like **count**, **photon**, **pixel** are always used in lower case and singular form.

The recommended way to indicate that there is no unit (for example a quantity that is a character string, or unitless) is to use an empty string rather than blanks or dashes.

The question mark (**?**) is reserved for cases when the unit is unknown.

2.7 Mathematical expressions containing symbols

Table 14 on page 25 summarizes how mathematical operations can be applied on unit symbols for exponentiation, multiplication, division, and other computations.

The combination rules is the point where the largest discrepancies between the different schemes appear. The FITS document summarises the problem of trying to best accomodate the existing schemes:

> Two or more base units strings (called str1 and str2 (...)) may be combined using the restricted set of (explicit or implicit) operators that provide for multiplication, division, exponentiation, raising arguments to powers, or taking the logarithm or square-root of an argument. Note that functions such as log actually require dimensionless arguments, so that log(**Hz**), for example, actually means log(x/1**Hz**). The final units string is the compound string, or a compound of compounds, preceded by an optional numeric multiplier of the form `10**k`, `10^k`, or `10±k` where `k` is an integer, optionally surrounded by parentheses with the sign character required in the third form in the absence of parentheses. Creators of FITS files are encouraged to use the numeric multiplier only when the available standard scale factors (. . .) will not suffice. Parentheses are used for symbol grouping and are strongly recommended whenever the order of operations might be subject to misinterpretation. A space character implies multiplication which can also be conveyed explicitly with an asterisk or a period. Therefore, although spaces are allowed as symbol separators, their use is discouraged. Note that, per IAU convention, case is significant throughout. The IAU style manual forbids the use of more than one slash (/) character in a units string. However, since normal mathematical precedence rules apply in this context, more than one slash may be used but is discouraged.

str1 str2	Multiplication (discouraged – see text)
str1*str2	Multiplication
str1.str2	Multiplication
str1/str2	Division
str1expr**	Raised to the power expr
str1^expr	Raised to the power expr
str1expr	Raised to the power expr
log(str1)	Common Logarithm (to base 10)
ln(str1)	Natural Logarithm
exp(str1)	Exponential (e^{str1})
sqrt(str1)	Square root

Table 7: Combination rules and mathematical expressions for VOUnits.

> A unit raised to a power is indicated by the unit string followed, with no intervening spaces, by the optional symbols `**` or `^` followed by the power given as a numeric expression (...). The power may be a simple integer, with or without sign, optionally surrounded by parentheses. It may also be a decimal number (e.g. 1.5, 0.5) or a ratio of two integers (e.g. 7/9), with or without sign, which must be surrounded by parentheses. Thus meters squared may be indicated by m**(2), `m**+2`, `m+2`, `m2`, `m^2`, `m^(+2)`, etc. and per meter cubed may be indicated by `m**-3`, `m-3`, `m^(-3)`, `/m3`, and so forth. Meters to the three-halves power may be indicated by `m(1.5)`, `m^(1.5)`, `m**(1.5)`, `m(3/2)`, `m**(3/2)`, and `m^(3/2)`, but not by `m^3/2` or `m1.5`. (Pence et al., 2010, §4.3.1)

This and other ambiguities are discussed in the detailed syntaxes of Appx. C.

VOUnits follow the same rules as FITS, as summarized in Table 7, and can in addition be preceded by an arbitrary optional numeric scaling factor (expressed as a floating number with the optional exponent being expressed as in FITS, with a multiplication symbol). Examples of valid VOUnits are **2.54cm**, **10+8m**, **3.45 10**(-4)Jy**, **10.6E-9g**. Scale factors from Table 10 on page 20 should however be preferred and used whenever possible.

2.8 Remarks and good practices

VOUnits, as described in this document, provide a string serialization of unit labels compatible with the two principal means of representation expected in the VO:

- as an attribute string in a VOTable document: `unit="..."`

- as a unit element in an XML serialization of some data model: `<unit>...</unit>`

Avoiding the use of special characters (Å, ', ", ...) reduces the complexity of using VOUnits in query languages or other programming environments.

Even if enclosing VOUnits with single or double quotes should be sufficient to facilitate parsing of a general query containing VOUnits, avoiding the use of white spaces for multiplication should make the parsing even easier, with the unit label being a single 'word'. Therefore the notation in the first line of Table 7 is discouraged.

Complex data formats are not addressed by VOUnits. While possibly convenient for humans, sexagesimal coordinates or calendar dates expressed in ISO 8601 are quantities represented in a complex format, encoded as strings, and as such the corresponding VOUnit should be an empty string. Expressions such as "d:m:s" or "ISO 8601" are not valid VOUnits. This should not be a problem, as existing VO standards already recommend that coordinates be expressed in decimal degrees.

One can also remark that some quantities like the Modified Julian Date (MJD) are not recognized VOUnits. As described in Sect. 1.2, the quantity MJD can be seen as a concept (described by the appropriate UCD or utype), and the corresponding value will most likely be expressed in days, so the VOUnit will be **d**. There is no need in overloading VOUnits to incorporate the description of concepts themselves.

A summary of the various symbols is given in the last part of Appendix B.

The notion of unit conversion and quantity manipulation is discussed in Sect. 3.3.

3 Use cases and applications

3.1 Unit parsing

The rules defined in Sect. 2 allow us to build VOUnit parsers. Several services can be built on top of a VOUnit parser:

1. Validation. A service checking that a VOUnit is well written. The output of such a service can have different levels: fully valid unit; valid syntax, but not the preferred one (e.g. use of deprecated symbols); parsing error.
2. Explanation. A service returning a plain-text explanation of the unit label.
3. Typesetting. A service returning an equivalent of the unit label suitable for inclusion in a LaTeX or HTML document.
4. Dimensional equation. As described by Osuna and Salgado (2005), VOUnits can be translated into a dimensional equation, allowing to build up conversions methods from one string representation to another one (see also Sect. 3.3).

3.2 Libraries

There are a few existing libraries able to interpret unit labels. In all cases, some software effort is required if they are to be used in translating between data provider unit labels, and those to be adopted by the IVOA for internal use.

One of the most widely-used specialised astronomical libraries is AST which includes a unit conversion facility attached to astronomical coordinate systems (Berry and Warren-Smith, 1997–2011).

Another library has been developed at CDS[2], and can be tested online[3]. This library covers all the symbols and notations defined in the standard for astronomical catalogues (CDS, 2000, §3.2), as well as additional symbols and notations.

The Unity library[4] is a new standalone library intended to parse VOUnits, OGIP, StdCats and FITS formats; it was used as a vehicle for developing and testing the grammars and ideas for this present document. It provides yacc-style grammars for the various syntaxes, as well as implementing them in parsers written in Java and C. The grammars of Appx. C are extracted from the Unity distribution.

3.3 Unit conversion and quantity transformation

Unit conversion is the simple task of converting a quantity expressed in a given unit into a different unit, while the concept remains the same. For example, converting a distance in pc into a distance in AU or km. Or converting a flux from mJy to W.m-2.Hz-1. This is rather easy with existing libraries, using dimensional analysis or SI units as a reference.

Quantity transformation consists in deriving a new quantity from one or several original quantities. It is more complex, because it requires to have a precise model (a simple equation in simple cases) for computing the transformation. The model involves quantities, each described with a UCD or utype, value and VOUnit. Some of the quantities involved might be physical constants (e.g. Boltzmann's constant k_B).

Examples of such transformations can be:

- linear unit conversion: a distance is measured in pixel in an image, and needs to be transformed in the corresponding angular separation in arcsec. This can be done if the quantity representing the pixel scale is given, with its value and a compatible unit like deg/pixel.
- converting a photon wavelength in the corresponding photon energy or frequency.

[2] http://cds.u-strasbg.fr/resources/doku.php?id=units
[3] http://cdsweb.u-strasbg.fr/cgi-bin/Unit
[4] https://bitbucket.org/nxg/unity

- deriving the flux for a given photon emission rate (in `W`) from Planck's constant (`6.63` 10^{-34} `J.s`), the radiation frequency (in `GHz`), and the number of photons emitted per second.
- transforming a magnitude into a flux, as needed for SED building.

VOUnits can help in quantity transformation if all quantities are qualified with proper VOUnits.

3.4 Query languages

Including VOUnits in queries is not an easy task. Some guidelines were defined in the reflexion on ADQL.

1. All data providers should be encouraged to supply units for each column of a table. Columns should also have associated UCDs, so that quantities can be properly identified.
2. The IVOA needs to provide a parser to relate the native units to the standard IVOA labels (in this context, the 'native units' are the units of the underlying database table or metadata).
3. The default response to a query which does not specify units, will be in the native units of the table.
4. 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[5], for instance. If any column used in responding to a query lacks a necessary unit, the output involving that column will be unitless.
5. 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?

3.5 Broader use in the VO

Different VO entities require and consume metadata with units attached like registries, applications and interoperate via protocols. Fig. 2 illustrates the places where the IVOA could intervene to ensure consistent use of units.

[5] http://saada.unistra.fr/

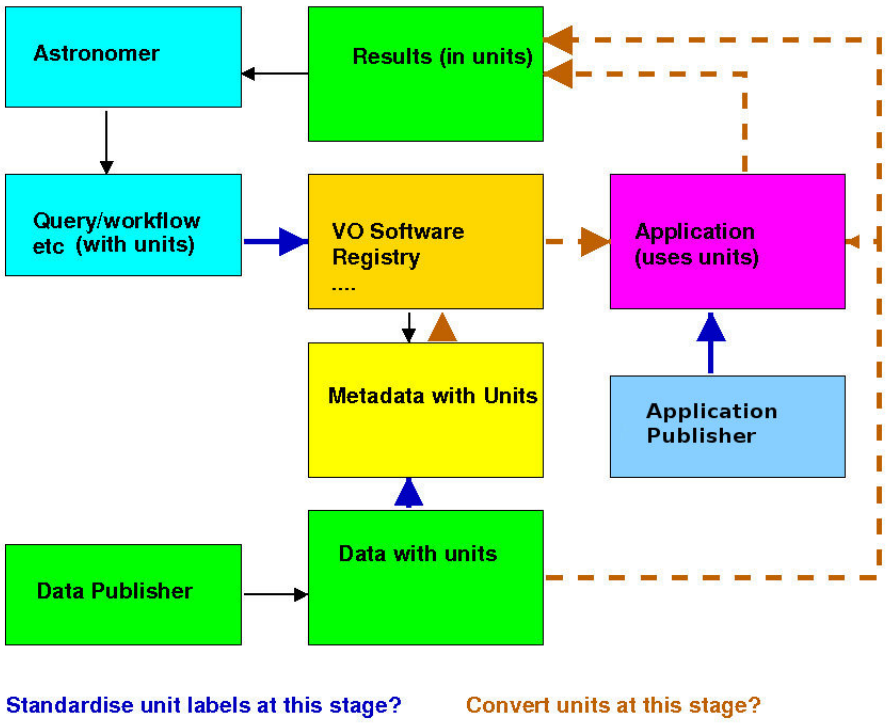

Figure 2: This shows the levels at which conversions might be done. Blue arrows: At the point where an astronomer or data provider submits input to the VO, we should provide tools to ensure that units are labeled consistently according to VOUnits. This implies that a units parsing step is included prior to metadata ingestion into the VO. 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.

A Current use of units

Many other projects have already produced lists of preferred representations of units. Those most commonly used in astronomy are described in this section.

The four first schemes described below are used as references for the comparison tables presented later in this document.

A.1 IAU 1989

In the section 5.1 of its Style Manual, the IAU gives a set of recommendations for representing units in publications (IAU Commission 5, 1989). This document therefore provides useful reference guidelines, but is not directly

applicable to VOUnits because the recommendations are more intended for correct typesetting in journals than for standardized metadata exchange. The IAU style will be summarized in the second column of the comparison tables.

A.2 OGIP 1993

NASA has defined a list of character strings specifying the basic physical units used within OGIP (Office of Guest Investigator Programs) FITS files (George and Angelini, 1995). Rules and guidelines on the construction of compound units are also outlined.

HEASARC datasets follow these conventions, presented in the third column of the comparison tables.

A.3 Standards for astronomical catalogues

The conventions adopted at CDS are summarized in the Standards for Astronomical Catalogues, Version 2.0 (CDS, 2000, §3.2). They are presented in the fourth column of the comparison tables.

A.4 FITS 2010

In Section 4.3 of the reference FITS paper, Pence et al. (2010) describe how unit strings are to be expressed in FITS files. The recommendations are presented in the fifth column of the comparison tables.

A.5 Other usages

- `http://arxiv.org/pdf/astro-ph/0511616`
  Dimensional Analysis applied to spectrum handling in VO context (Osuna and Salgado, 2005) offers a mathematical framework to guess and recompute SI units for any quantity in astronomy.
- `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
- `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.)

B Comparison of existing schemes and VOUnits

	IAU	OGIP	StdCats	FITS	VOUnits
Units are strings of chars		YES		YES	YES
Case sensitive	YES	YES	YES	YES	YES
Character set			No spaces	ASCII text	ASCII printable

Table 8: Comparison of string representation and encoding.

	IAU	OGIP	StdCats	FITS	VOUnits
The 6+1 base SI units (use s, not sec, for seconds)	m, s, A, K, mol, cd				idem
	unit is kg, but use g with prefixes	kg	g	kg, note that g is allowed	g
Dimensionless planar and solid angle	rad, sr				idem
				deg preferred for decimal angles	
Derived units with symbols	Hz, N, Pa, J, W, C, V, S, F, Wb, T, H, lm, lx				idem
	Ω	ohm	Ohm	Ohm	Ohm

Table 9: Comparison of base units.

	IAU	OGIP	StdCats sec. 3.2.3	FITS	VOUnits
Scale factors, (multiple) prefixes	d, c, m, n, p, f, a da, h, k, M, G, T, P, E μ	u z, y, Z, Y			idem u z, y, Z, Y
Prefix–symbol concatenation	no space, regarded as single symbol	no space, regarded as a single unit string	no space	no space (implicit)	no space
Prefix-able symbols	Not kg: use g	All units above, and eV, pc, Jy, Crab Only mCrab allowed	all	all	all (except P for a)
Use compound prefixes	should not	should never	must not	must not	must not

Table 10: Comparison of scale factors.

	IAU	OGIP	StdCats	FITS	VOUnits
minute	min, [m]	min	min	min	min
hour	h, [h]	h	h	h	h
day	d, [d]	d	d	d	d
year	a	yr	a, yr	a, yr Pa (peta a) forbidden	like FITS
arcsecond	″	arcsec	arcsec	arcsec	arcsec
arcminute	′	arcmin	arcmin	arcmin	arcmin
degree (angle)	°	deg	deg	deg	deg
milliarcsecond	mas (use nrad!)		mas	mas	mas
microarcsec			uarcsec		no dedicated symbol, use uarcsec
cycle	c, [c]				not used
astronomical unit	au	AU	AU	AU	AU
parsec	pc				pc
atomic mass	u			u	u
electron volt	eV				eV
jansky	Jy				Jy
celsius degree	°C for meteorology, other use K				not used
century	ha, cy should not be used				no dedicated symbol, use ha or hyr

Table 11: Comparison of astronomy-related units.

	IAU	OGIP	StdCats	FITS	VOUnits
ångström	Å	angstrom	0.1nm	Angstrom	angstrom, Angstrom
micron	μ				not used
fermi	no symbol				not used
barn	b	barn	barn	barn	barn
cubic centimetre	cc				no dedicated symbol
dyne	dyn				not used
erg	erg	erg	No symbol. mW/m2 used for erg.cm-2.s-1	erg	erg
calorie	cal				not used
bar	bar				not used
atmosphere	atm				not used
gal	Gal				not used
eotvos	E				not used
gauss	G	G		G	G
gamma	γ				not used
oersted	Oe				not used
Imperial, non-metric	should not be used				not used

Table 12: Comparison of symbols deprecated by IAU.

	IAU	OGIP	StdCats	FITS	VOUnits
magnitude		mag			mag
rydberg			Ry	Ry	
solar mass	$M_\odot$		solMass	solMass	
solar luminosity			solLum	solLum	
solar radius			solRad	solRad	
light year		lyr		lyr	
count		count	ct	ct, count	
photon		photon		photon, ph	same as FITS
rayleigh				R	
pixel		pixel	pix	pix, pixel	
debye			D	D	
relative to Sun			Sun	Sun	
channel		chan		chan	
bin		bin		bin	
voxel		voxel		voxel	
bit			bit	bit	
byte		byte	byte	byte	
adu				adu	
beam				beam	
		Crab avoid use			not used
No unit, dimensionless		blank string	-		empty string
Unitless in percent			%		
unknown		UNKNOWN			?

Table 13: Comparison of other symbols.

	IAU	OGIP	StdCats	FITS
Multiplication	space, except if previous unit ends with superscript; dot (.) may be used	one or more spaces OR one asterisk (*) with optional spaces on either side	dot (.), no space	single space OR asterisk (*, no spaces) OR dot (., no spaces)
Division	per. Use negative index or solidus (/)	slash (/) with optional spaces on either side, space not recommended after / OR negative index	/ with no spaces	/ with no spaces
Use of multiple /	MUST never use two /	allowed	allowed	may be used, discouraged, math precedence rule
sym raised to the power y	superscript	sym**(y) parenthesis optional if $y > 0$	nothing: sym$y$ use +/- sign for 10+21	sym$y$ OR sym**(y) OR sym^(y), no space
Exponential of sym		exp(sym)		exp(sym)
Natural log of sym		ln(sym)		ln(sym)
Decimal log of sym		log(sym)	[sym]	log(sym) dimensionless argument

Square root of sym		sqrt(sym)		sqrt(sym)
Other math		sin(sym), cos(sym), tan(sym), asin(sym), acos(sym), atan(sym), sinh(sym), cosh(sym), tanh(sym)		not used
( )		any allowed	allowed	optional around powers
powers	superscripts	decimal and integer fractions allowed	integers only	integer (sign and () optional), OR decimal or ratio between ()
Numeric factor	not used	should be avoided; only powers of 10 allowed; should precede any unit string	allowed	optional 10**k, 10^k, or 10±k

Table 14: Comparison of mathematical expressions and symbols combinations.

C Formal grammars

In this section we provide formal (yacc-style) grammars for the four ASCII-based syntaxes discussed in this document. The FITS, OGIP and CDS grammars are not normative: the corresponding specification documents do not provide grammars, and instead describe the syntaxes in text, so that the grammars here are deductions from the specification text. This unfortunately means that some of these syntaxes are ambiguous. These ambiguities

`CARET`	the `^` character
`DIVISION`	the solidus, `/`
`DOT`	the dot/period/full-stop character
`FLOAT`	a string matching the regular expression `[-+]?[0-9]+\.[0-9]+`
`LIT10`	a literal string '`10`' (without quotes).
`OPEN_P` / `CLOSE_P`	parentheses
`SIGNED_INTEGER`	an integer with a required leading sign
`STAR`	the asterisk
`STARSTAR`	a pair of asterisks, `**`
`STRING`	a sequence of letters (a–z and A–Z) plus the percent sign
`UNSIGNED_INTEGER`	an integer with no leading sign
`WHITESPACE`	a non-empty string of space characters (no other whitespace)

Table 15: The terminals used in the grammars

are discussed in the sections below. We recommend that VO applications parse these syntaxes in a way which is consistent with the grammars here. The grammar for the VOUnits syntax, in Appx. C.4, is normative.

We believe that the grammars below are such that if a string successfully parses in two distinct grammars, it means the same in both.

The grammars here are those of the 'Unity' package at `https://bitbucket.org/nxg/unity`, which includes lists of recommended base and derived units in the various syntaxes, plus a collection of test cases.

In these grammars, the terminals are as given in Table 15.

C.1 FITS

For the FITS units syntax, see section 4.3 of Pence et al. (2010), and its associated tables. Our preferred FITS grammar is in Table 16 on the next page.

As noted above in Sect. 2.7, the FITS specification isn't completely clear on the topic of solidi, saying '[t]he IAU style manual forbids the use of more than one solidus (/) character in a units string. However, since normal mathematical precedence rules apply in this context, more than one solidus may be used but is discouraged'. This does not really resolve the question of whether, for example, `kg/m s` should be parsed as $\mathrm{kg\,m^{-1}\,s^{-1}}$ or as $\mathrm{kg\,m^{-1}\,s}$, since this is a question of both operator precedence and (left-)associativity, where there might be different rules internationally, and conflicts between mathematical and programming-language rules. Most people would *probably* parse it as $\mathrm{kg\,m^{-1}\,s^{-1}}$, but we trust that most educators would oblige students to rewrite the expression on the grounds that any ambiguity is too

```
input:            product_of_units
                | scalefactor product_of_units
                | scalefactor WHITESPACE product_of_units
                | product_of_units division unit_expression
                | scalefactor product_of_units division unit_expression
                | division unit_expression

product_of_units: unit_expression
                | product_of_units product unit_expression

unit_expression:  unit
                | OPEN_P product_of_units CLOSE_P

scalefactor:      LIT10 power numeric_power
                | LIT10 SIGNED_INTEGER

division:         DIVISION

unit:             STRING
                | STRING power numeric_power
                | STRING numeric_power

power:            CARET
                | STARSTAR

numeric_power:    integer
                | OPEN_P integer CLOSE_P
                | OPEN_P FLOAT CLOSE_P
                | OPEN_P integer division UNSIGNED_INTEGER CLOSE_P

integer:          SIGNED_INTEGER | UNSIGNED_INTEGER

product:          WHITESPACE | STAR | DOT
```

Table 16: The FITS grammar

much. Here, we resolve the ambiguity by declaring that there can be only a single expression to the right of the solidus.

It is a consequence of this that nothing can be successully parsed in two different grammars, with different meanings. If the right-hand-side of the division could be a `product_of_units`, then `kg /m s` would parse in both the FITS and OGIP syntaxes, but mean $\mathrm{kg\,m^{-1}\,s^{-1}}$ in the FITS syntax, and $\mathrm{kg\,m^{-1}\,s}$ in the OGIP one.

Other ambiguities:

- The FITS specification may or may not be intended to permit `10+3 /m`, but we don't.
- It is possible to read the FITS spec as permitting `m^1.5`, without parentheses. We take it to be invalid here.

C.2 OGIP

For the OGIP units syntax, see George and Angelini (1995). Our preferred OGIP grammar is in Table 17.

```
input:            product_of_units
                | scalefactor product_of_units
                | scalefactor WHITESPACE product_of_units

product_of_units: unit_expression
                | division unit_expression
                | product_of_units product unit_expression
                | product_of_units division unit_expression

unit_expression:  unit
                | OPEN_P product_of_units CLOSE_P

scalefactor:      LIT10 power numeric_power
                | LIT10
                | FLOAT

division:         DIVISION | WHITESPACE DIVISION
                | WHITESPACE DIVISION WHITESPACE | DIVISION WHITESPACE

unit:             STRING
                | STRING power numeric_power

power:            STARSTAR

numeric_power:    UNSIGNED_INTEGER
                | FLOAT
                | OPEN_P integer CLOSE_P
                | OPEN_P FLOAT CLOSE_P
                | OPEN_P integer division UNSIGNED_INTEGER CLOSE_P

integer:          SIGNED_INTEGER | UNSIGNED_INTEGER

product:          WHITESPACE | STAR | WHITESPACE STAR
                | WHITESPACE STAR WHITESPACE | STAR WHITESPACE
```

Table 17: The OGIP grammar

Specification ambiguities:

- The OGIP specification permits a space between the leading factor and the rest of the unit (by implication from the provided examples).
- OGIP *recommends* having no whitespace after the division solidus, but does not forbid it; therefore we permit it in this grammar.
- From its specification text, OGIP appears to permit `str1**y`, where `y` can be a float, even though none of its examples include this. The same interpretive logic would appear to permit `m**3/2`, but this seems

to run too great a risk of being misparsed, and we forbid it here.

- In the same place, the text suggests that `str1**y` may omit the brackets 'if `y` is positive', but the context suggests that the intention is to permit this if `y` is unsigned. In the grammar here, we permit the omission of the brackets only if `y` is unsigned – that is, `m**+2`, like `m**-2`, is forbidden.

C.3 The CDS grammar

For the CDS units syntax, see (CDS, 2000, §3.2). Our preferred CDS grammar is in Table 18.

```
input:            product_of_units
                | scalefactor product_of_units

product_of_units: unit_expression
                | division unit_expression
                | product_of_units product unit_expression
                | product_of_units division unit_expression

unit_expression:  unit
                | OPEN_P product_of_units CLOSE_P

scalefactor:      LIT10 power numeric_power
                | LIT10 SIGNED_INTEGER
                | UNSIGNED_INTEGER
                | LIT10
                | CDSFLOAT
                | FLOAT

division:         DIVISION

unit:             STRING
                | STRING numeric_power

power:            STARSTAR

numeric_power:    integer

integer:          SIGNED_INTEGER | UNSIGNED_INTEGER

product:          DOT
```

Table 18: The CDS grammar

The `CDSFLOAT` terminal is a string matching the regular expression

```
[0-9]+\.[0-9]+x10[-+][0-9]+
```

(that is, something resembling `1.5x10+11`).

C.4 The VOUnits grammar

The VOUnits grammar is defined by this section, the grammar in Table 19 (with the terminals of Table 15 on page 26) and the known units of Table 1 on page 8. This grammar is a strict subset of the FITS and CDS grammars (in the sense that any VOUnit unit string is a valid FITS and CDS string, too), and it is almost a subset of the OGIP grammar, except that it uses the dot for multiplication rather than star. By design, therefore, a unit specification written to conform with the VOUnits grammar is immediately readable (with the same semantics) by a FITS or CDS parser, and almost readable by an OGIP one.

In particular:

- The product of units is indicated only by a dot, with no whitespace: `N.m`.
- Raising a unit to a power is done only with a double-star: `kg.m**2.s**-2`.
- There may be at most one division sign at the top level of an expression.

```
input:            product_of_units
                | scalefactor product_of_units
                | product_of_units division unit_expression
                | scalefactor product_of_units division unit_expression
                | division unit_expression

product_of_units: unit_expression
                | product_of_units product unit_expression

unit_expression:  unit
                | OPEN_P product_of_units CLOSE_P

scalefactor:      LIT10 power numeric_power

division:         DIVISION

unit:             STRING
                | STRING power numeric_power

power:            STARSTAR

numeric_power:    integer
                | OPEN_P integer CLOSE_P
                | OPEN_P FLOAT CLOSE_P
                | OPEN_P integer division UNSIGNED_INTEGER CLOSE_P

integer:          SIGNED_INTEGER | UNSIGNED_INTEGER

product:          DOT
```

Table 19: The VOUnits grammar

D Updates of this document

- 1.0-20130701: Simplified Architecture diagram. Added example with scientific notation. Kept grammar tables location Appendix C pages.
- 1.0-20130419: Some restructuring, some rephrasing, and a few layout changes.
- 1.0-20121212 to 1.0-20130225: Large tables from section 3 moved to Appendix A. Short summaries of symbols added to section 3. Changes to table of known units for consistency with text. Added explanations for units Sun and byte.
- 1.0-20120820 to 1.0-20121212: Minor typographical fixes. Added definition of OGIP. Removed last sentence from acknowledgements, which have been moved to the beginning of the document. Changed figure 1 to move Units in Semantics. Added 'discouraged' in first line of Table 7. Color change in figure 2 and its label.
- 1.0-20120718 to 1.0-20120801: Minor typographical fixes
- 1.0-20120718 to 1.0-20120801:
  - Included yacc-style grammars in document.
- 1.0-20120521 to 1.0-20120718:
  - Removed external tables refs in tables to avoid confusion.
  - Removed refs to SOFA and NOVAS.
  - Precision on the "no unit" case in text.
  - Added formal grammar in annex.
  - Minor editing and typo fixes.
- 1.0-20111216 to 1.0-20120521:
  - Typos fixed, removed F. Bonnarel from authors.
  - One sentence rephrased in section 1.2 for clarity.
  - Clarification of g and kg issue in Sect. 2.3.
  - Added remark on Pa in Sect. 2.4.
  - Micro-arcsecond and century explained in Table 11 on page 21.
  - Table 12 on page 22 completed.
  - Added numeric factors in Table 14 on page 25 and discussion in text.
- 0.3 to 1.0-20111216: Major rework of the document.

References

David S Berry and Rodney F Warren-Smith. AST - a library for handling world coordinate systems in astronomy. Technical report, Starlink Project, 1997–2011. URL http://www.starlink.ac.uk/ast.

BIPM. *Le Système international d'unités / The International System of Units ('The SI Brochure')*. Bureau international des poids et mesures, eighth edition, 2006. URL http://www.bipm.org/en/si/si_brochure/.

CDS. Standards for documentation of astronomical catalogues. Technical report, Centre de Données astronomiques de Strasbourg, February 2000. URL http://cds.u-strasbg.fr/doc/catstd.htx.

Ian M George and Lorella Angelini. Specification of physical units within OGIP FITS files. Web page, May 1995. URL http://heasarc.gsfc.nasa.gov/docs/heasarc/ofwg/docs/general/ogip_93_001/.

IAU Commission 5. The IAU style manual: The preparation of astronomical papers and reports. In George A Wilkins, editor, *Transactions of the IAU*, volume XX B. Kluwer, 1989. ISBN 0-7923-0550-7. URL http://www.iau.org/static/publications/stylemanual1989.pdf.

IEC 80000-13. Quantities and units —- part 13: Information science and technology (iec 80000-13:2008). International Standard, 2008. Replaces sections 3.8 and 3.9 of IEC-60027-2 (binary prefixes); see also IEEE-1541.

François Ochsenbein, Roy Williams, Clive Davenhall, Daniel Durand, Pierre Fernique, David Giaretta, Robert Hanisch, Thomas McGlynn, Alex Szalay, Mark B. Taylor, and Andreas Wicenec. VOTable format definition version 1.2. IVOA Recommendation, October 2011, arXiv:1110.0524.

Pedro Osuna and Jesus Salgado. Dimensional analysis applied to spectrum handling in virtual observatory context, November 2005, arXiv:astro-ph/0511616.

William D Pence, L. Chiappetti, Clive G Page, R. A. Shaw, and E. Stobie. Definition of the Flexible Image Transport System (FITS), version 3.0. *Astronomy and Astrophysics*, 524:A42+, December 2010. doi: 10.1051/0004-6361/201015362.